UNIVERSIDADE DO MINDELO

DEPARTAMENTO DE CIÊNCIAS ECONÓMICAS E EMPRESARIAIS

CURSO DE LICENCIATURA EM INFORMÁTICA DE GESTÃO

RELATÓRIO DE PROJETO DE LICENCIATURA

ANO LECTIVO 2014/2015 – 4º ANO

Autor: Leonildo Lopes Gomes, N.º 2124

Mindelo, 2015

Projecto de Licenciatura em Informática de Gestão

Leonildo Lopes Gomes

TÍTULO:

SOS MINDELO - APLICAÇÃO MÓVEL DE APOIO A SERVIÇOS DE EMERGÊNCIAS DE SAÚDE

Trabalho apresentado à **Universidade do Mindelo** como parte dos requisitos para obtenção do grau de licenciatura em Informática de gestão

Orientador:
Mestre João do Monte Gomes Duarte

Mindelo, 2015

Resumo

O Android é um sistema operativo de código fonte aberto desenvolvido com foco principal nos dispositivos móveis. Este trabalho pretende aplicar os recursos disponíveis no Android, nomeadamente os serviços de geolocalização (GPS), API do Google Maps, suporte a dados, entre outros para desenvolver uma aplicação móvel de apoio a serviços de emergências de saúde na cidade do Mindelo denominada de SOS Mindelo.

A aplicação SOS Mindelo é uma ferramenta que permite localizar as farmácias, hospitais, clínicas e centros de saúde que estejam mais próximos geograficamente da posição do utilizador no momento da utilização da aplicação.

As classes que compõe o código fonte da aplicação foram implementadas com uso da linguagem de programação JAVA enquanto que para as interfaces de utilizador se baseou na linguagem XML. A base de dados foi desenvolvida com recurso às linguagens PHP e MySql e o mesmo se encontra alojado na internet no servidor *online* Hostinger. Para a interligação entre a base de dados remota e a aplicação instalada no dispositivo do utilizador se recorreu á ferramenta Java Script Object Notation. Para a estimação da distância e a selecção de rotas entre a posição do dispositivo do utilizador e os possíveis destinos se recorreu ao API do Google Maps para além da implementação de um método para o cálculo da distância euclidiana entre dois pontos.

Tendo em conta os parcos recursos que caracterizam os dispositivos móveis em termos de poder de processamento, espaço de armazenamento e eficiência energética as imagens, textos e outros recursos são guardados na base de dados *online* e acessados pelo terminal do utilizador quando necessário.

O presente relatório está organizado da seguinte forma: na introdução são descritos os objectivos do desenvolvimento desta aplicação, bem como as vantagens da sua implementação. O primeiro capítulo apresenta a fundamentação teórica e as tecnologias utilizadas para o desenvolvimento da aplicação. O segundo capítulo descreve o processo de desenvolvimento da aplicação, nomeadamente a especificação dos requisitos, a apresentação dos diagramas de casos de usos e de classes, a implementação da base de dados remota, as implementações em código e as correspondentes interfaces de utilizador. Finalmente serão apresentados as conclusões finais do projecto e as perspectivas para trabalhos futuros.

Palavras-Chave: Android, GPS, Java, JSON, Activity, MySQL

ABSTRACT

Android is an open operating system developed with its main focus on mobile devices. This work intends to apply the resources available on Android, including geolocation services (GPS), Google Maps API, Data Support, and so on, for developing a mobile application called SOS Mindelo to provide assistance to users in case of health service emergencies in the city of Mindelo.

The SOS Mindelo application allows users to locate the pharmacies, hospitals, clinics and health centers that are geographically closer to their position at the time of the search.

The source code of this application was implemented in terms of classes using the Java programming language while the graphical user interfaces were implemented using the XML programming language. The database was developed using the PHP and MySql languages and it is hosted in the internet at the Hostinger online server. For the interconnection between the remote database and the application running on user's device the Java Script Object Notation tool was employed. For the calculation of distances and the selection of routes between users and the possible destinations the Google Maps API was used and a method for calculating the Euclidean distance between two points were also implemented.

Due to the limitations of the mobile devices in terms of resources such as processing power, storage and energy efficiency the images, texts and other resources are stored in the online database and accessed by the user's terminal when needed.

This report is organized as follows: the introduction part describes the objectives of the development of this application as well as the advantages of its implementation. The first chapter presents the theoretical foundation and the technologies used to develop this application. The second chapter describes the application development process, including the specification of the requirements, use cases' diagrams and classes' diagrams, the implementation of the remote database, the source code development and the corresponding user interfaces. Finally the conclusions and insights for future works are also presented.

Keywords: Android, GPS, Java, JSON, Activity, MySQL

Agradecimento

Agradeço primeiramente a Deus pela força e coragem para a realização deste trabalho.

Aproveito esta oportunidade para agradecer também a todos os meus familiares, em especial aos meus pais e aos meus irmãos, pelo apoio incondicional durante toda a minha formação.

A minha noiva, pelo carinho, amor e atenção durante essa trajectória.

A todos os professores, amigos e colegas de turma com quem convivi nesses anos de curso.

Ao meu orientador Mestre João do Monte Gomes Duarte pelo apoio, disponibilidade e paciência durante o desenvolvimento deste projecto.

A todos, um especial obrigado.

Indice

Lista de Abreviaturas

API - Application Programming Interface

GPS - Global Positioning System

PHP - Hypertext Preprocessor

JSON - Java Script Object Notation

IDE - Integrated Development Environment

SDK - Software Development Kit

ADT - Android Development Tools

XML - Extensible Markup Language

HTTP - Hypertext Transfer Protocol

IDE - Integrated Development Environment

JDK - Java Development Kit

AVD - Android Virtual Device

SGBD - Sistema de Gestão de Base de Dados

JAR - Java Archive

IOS - iPhone Operating System (Apple)

ADB - Android Debug Bridge

PDT - PHP Development Tools

GSM - Global System for Mobile Communications

USB - Universal Serial Bus

HD - High Definition

IDC - International Data Corporation

Índice de Figuras

Lista de Equação

Lista de Tabela

Lista de Gráficos

Introdução

Com o surgimento e a evolução dos dispositivos móveis, os mesmos passaram a fazer parte do dia-a-dia das pessoas, trazendo uma série de vantagens. Este novo cenário levou a uma mudança de paradigma no respeitante ao acesso a dados e serviços disponibilizados via Cloud [(1)]. Correntemente, importantes serviços, como o acesso a internet, manuseamento de contas bancárias, pagamento de facturas electrónicas, entre outros, são efectuados com uso de *smartphones*, *tablets* e outros dispositivos. E estas práticas já fazem parte do quotidiano cabo-verdiano.

Com a introdução da tecnologia 3G em Cabo Verde, verifica-se a aplicação desta nos dispositivos móveis, nomeadamente *smartphones* com sistemas operativos Android e IOS [(2)] [(3)]. Neste sentido, torna-se necessário o desenvolvimento de aplicações orientadas a suprir as necessidades emergentes deste novo sector, que tem como vantagem fundamental o facto de os utilizadores terem acesso aos seus dados e serviços permanentemente.

Com o desenvolvimento da aplicação SOS Mindelo, pretende-se ainda levar em conta as limitações dos dispositivos móveis em termos de velocidade, processamento, espaço de armazenamento e alimentação quando comparados com o desenvolvimento de aplicações em ambientes tradicionais.

Objectivo Principal

Este trabalho tem como objectivo principal desenvolver o SOS Mindelo, uma aplicação para dispositivos móveis com sistema operativo Android, que sirva de apoio aos utilizadores na área de saúde e seja uma ferramenta útil em caso de emergências médicas.

Objectivos Específicos

- Localizar todas as farmácias, hospitais, clínicas e centros de saúde da cidade do Mindelo mais próximas do utilizador no momento solicitado, utilizando a localização geográfica do dispositivo;
- Informar o utilizador sobre a localização, horário de funcionamento, contacto e o director farmacêutico das farmácias;
- Informar sobre as farmácias de serviço da cidade do Mindelo;
- Informar o utilizador sobre a localização e o contacto do hospital;
- Informar o utilizador sobre a localização, horário de funcionamento, director técnico e o contacto das clínicas;
- Informar o utilizador sobre as especialidades de cada clinica;
- Informar o utilizador sobre a localização, horário de funcionamento e o contacto dos centros de saúde;
- Exibir as entidades de emergências da cidade do Mindelo.
- Determinar no mapa a localização dos lugares presentes na aplicação e do utilizador;
- Exibir no mapa, uma rota de destino entre o utilizador e o lugar escolhido tendo em conta as suas posições geográficas;
- Efectuar ligações telefónicas aos locais presentes na aplicação com uso da própria aplicação;

- Envio de email ás entidades de emergências com uso da própria aplicação;
- Efectuar chamadas para números de emergências;

Motivação

A procura da informação é uma tarefa com que as pessoas lidam diariamente, e cada vez mais os utilizadores pretendem ter acesso às informações com maior ubiquidade possível. O aumento da utilização dos dispositivos móveis em Cabo Verde, alheado a falta de meios eficientes para a apresentação de informações nomeadamente na área de saúde foram as principais motivações que levaram ao desenvolvimento da aplicação SOS Mindelo.

Com esta aplicação os utilizadores possuem uma ferramenta capaz de disponibilizar informações de todas as farmácias, hospitais, clínicas e centros de saúde da cidade do Mindelo de forma permanente, permitindo que os utilizadores visualizem uma lista de farmácias, clínicas, hospitais, centros saúde mais próximas do utilizador no momento solicitado. Através desta lista o utilizador tem a possibilidade de visualizar informações com maior detalhe, tendo ainda a possibilidade de realizar uma chamada directamente da aplicação ao referido lugar e ainda visualizar uma rota, no mapa, para chegar a esse local.

Uma outra funcionalidade desta aplicação é de apresentar ao utilizador uma lista de contactos úteis permitindo ao utilizador realizar chamadas para as entidades de emergências e envio de email com um link, onde se pode ver no mapa a localização do utilizador.

Metodologia

Para a realização deste projecto, inicialmente foram realizadas pesquisas sobre as principais técnicas e ferramentas de desenvolvimento para a plataforma Android, as tecnologias do Google Maps API e os serviços de geolocalização.

Foram também efectuadas pesquisas sobre trabalhos relacionados com este tema que já tinham sido realizados, de onde se constatou a necessidade do desenvolvimento desta aplicação;

Seguidamente foi analisado o meio físico, onde foram recolhidos os dados de todas as farmácias, clinicas, hospitais e centros de saúde, da cidade do Mindelo.

Em seguida foi implementado uma base de dados, hospedada *online*, onde foram guardados os dados recolhidos. Para comunicação dos dados entre a aplicação Android e a base de dados *online*, foi criado uma *Web Service*.

Seguidamente foi desenvolvido uma aplicação para dispositivos com sistema operativo Android, englobando as tecnologias de Google Maps API e GPS integrado nos dispositivos Android.

Trabalhos Relacionados

A nível mundial existem várias aplicações com características similares ao desenvolvido neste projecto, de entre os quais, se podem destacar:

- Farmácias de Serviço.net;
- Emergency Medical Southtyrol;
- Pharmacies de Garde;
- Farmaguia;
- Pharmacies de garde Maroc;

A **Farmácia de Serviço.net** (4) é uma aplicação para dispositivos com sistema operativo Android, capaz de listar as farmácias de serviço em Portugal, com as respectivas localizações e contactos.

Fazendo uma comparação entre a aplicação SOS Mindelo e a aplicação acima referida, pode-se verificar que ambos possuem as seguintes funcionalidades comuns:

- Apresentam ao utilizador uma lista de farmácias mais perto do utilizador tendo em conta a localização do dispositivo;
- Permitem visualizar informações das farmácias com mais detalhes;
- Permitem entrar em contacto directamente com as farmácias através da aplicação;
- Permitem localizar as farmácias no mapa.
- Permitem obter uma rota de destino entre utilizador e o lugar escolhido.

A aplicação SOS Mindelo apresenta as seguintes vantagens em relação a aplicação Farmácia de Serviço.net:

- Para além das farmácias, apresenta informações de hospitais, clínicas, centros de saúde e contactos úteis;
- Permite realizar chamadas para números de emergências;
- Permite enviar email para as entidades de emergências;

A aplicação SOS Mindelo apresenta como desvantagem, em relação a aplicação acima referido o facto de abranger apenas a cidade do Mindelo, no entanto pode, num trabalho futuro ser alargada a todo o território nacional.

O **Emergency Medical Southtyrol** [5] é uma aplicação destinada a dispositivos móveis com sistema operativo Android, que serve de suporte em caso de emergências médicas no sul do Tirol (Itália).

Fazendo uma comparação entre esta aplicação e a aplicação SOS Mindelo, pode-se verificar que ambos possuem as seguintes funcionalidades comuns:

- Conseguem listar farmácias, hospitais mais próximos do utilizador tendo em conta a localização de dispositivo;
- Permitem visualizar informações dos locais com mais detalhes;
- Usam o mapa para determinar a localização dos locais acima referidos.
- Permitem a realização de chamadas para números de emergências a partir da aplicação;

A aplicação SOS Mindelo apresenta as seguintes vantagens em relação a aplicação acima referida:

- Permite para além de farmácias e hospitais, listar clínicas e centros de saúde;
- Permite o envio de um email com um link para as entidades de emergências, onde através de um clique se pode ver no mapa a localização do dispositivo;
- Permite ver no mapa a localização do dispositivo e o local escolhido e determinar uma rota de destino.

A aplicação SOS Mindelo apresenta como desvantagem o facto de não permitir pesquisar especificamente os lugares presentes na aplicação.

Pharmacies de Garde [6] é uma aplicação destinada a dispositivos com sistema operativo Android capaz de listar farmácias, hospitais, clinicas, médicos das principais cidades de Marrocos.

Fazendo uma comparação entre esta aplicação e a aplicação SOS Mindelo, pode-se verificar que ambos possuem as seguintes funcionalidades comuns:

- Conseguem listar farmácias, hospitais, clinicas;
- Permitem visualizar informações dos locais mais detalhadamente;
- Permitem realizar chamadas para números de emergências directamente da aplicação;

Relativamente à aplicação acima referenciada, a aplicação SOS Mindelo apresenta as seguintes vantagens:

- Permite o envio de um email com um link para as entidades de emergências, onde através de um clique se pode ver no mapa a localização do dispositivo;
- Apresenta os lugares presentes na aplicação mais próximos do utilizador, no momento solicitado de acordo com posição geográfica do dispositivo.

A aplicação SOS Mindelo apresenta as seguintes desvantagens:

- Abrange apenas a cidade do Mindelo, mas pode no futuro ser ampliada a todo o território;
- Não permite pesquisar especificamente os lugares presentes na aplicação;

A **Farmaguia** [(7)] também é uma aplicação para dispositivos móveis com o sistema Operativo Android, que permite procurar farmácias mais próximas do utilizador na província de Barcelona. Em comparação com a aplicação SOS Mindelo, pode-se verificar que ambos detêm as seguintes funcionalidades:

- Permitem ver as farmácias mais próximas do utilizador no momento solicitado;
- Permitem visualizar informações das farmácias com mais detalhes;
- Permitem ver quais as farmácias que estão de serviço;
- Permite ver no mapa a localização do dispositivo e a farmácia escolhida e determinar uma rota de destino.

Tendo feito as comparações pode-se verificar as seguintes vantagens da aplicação SOS Mindelo em relação a aplicação acima referida:

- Para além das farmácias, apresenta informações de hospitais, clínicas, centros de saúde e contactos úteis;
- Realiza chamadas para números de emergências;
- Permite o envio de um email com um link para as entidades de emergências, onde através de um clique se pode ver no mapa a localização do dispositivo.

A aplicação SOS Mindelo apresenta as seguintes desvantagens:

- Não apresenta os serviços oferecidos por cada farmácia;
- Não permite a pesquisa avançada das farmácias;

Pharmacies de garde Maroc [8] é uma aplicação móvel para dispositivos com sistema operativo Android, que permite ao utilizador todas as farmácias existentes em sete cidades do Marrocos. Fazendo uma comparação com a aplicação SOS Mindelo, pode-se verificar que ambos possuem funcionalidades semelhantes, nomeadamente:

- Permitem ver as farmácias que estão serviços;
- Permitem ver uma lista de farmácias;
- Permitem ver a rota de acordo com a posição do dispositivo e da farmácia pretendida;

Tendo feito as comparações pode-se verificar as seguintes vantagens da aplicação SOS Mindelo em relação a aplicação acima referida:

- Para além das farmácias, apresenta informações de hospitais, clínicas, centros de saúde e contactos úteis;
- Permitem ver com mais detalhes informações das farmácias;
- Realiza chamadas para números de emergências;
- Permite o envio de um email com um link para as entidades de emergências, onde através de um clique se pode ver no mapa a localização do dispositivo.

A aplicação SOS Mindelo apresenta as seguintes desvantagens:

- Abrange apenas a cidade do Mindelo, no entanto a aplicação pode ser alargada a todo o território nacional;
- Não permite adicionar e editar uma farmácia na própria aplicação.

1 A Plataforma Android

1.1 O Android

O Android é um sistema operativo livre de código aberto, baseado no sistema operativo Linux [(9)]. O Android é direccionado principalmente aos dispositivos móveis como Smartphones e Tablets.

A 5 de Novembro de 2007, a empresa Google anunciou ao público o Android e a criação da Open Handset Alliance (OHA) [(10)], um consórcio constituído por diversas empresas do mercado de telemóveis, com o objectivo de criar padrões de telefonia aberta. Entre as empresas participantes estão envolvidas o Google, Samsung, Motorola, Intel, LG, T-Mobile, HTC, Dell, ZTE Corporation.

Além do sistema operativo Android, o Google anunciou também aos programadores o Android SDK, que incluem ferramentas de desenvolvimento e API´s necessários para desenvolver aplicações para a plataforma Android, usando a linguagem de programação Java.

Actualmente, estão disponíveis diversas plataformas móveis no mercado, nomeadamente IOS, Windows Mobile, Blackberry OS, além de outros. Segundo a IDC [(11)], o Android domina o mercado de smartphones com uma quota de mercado de 76.6%, no quarto trimestre de 2014. Na tabela 1 se pode verificar os dados referentes a quota de mercado das plataformas móveis mais usadas no mundo.

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q4 2014	76.6%	19.7%	2.8%	0.4%	0.5%
Q4 2013	78.2%	17.5%	3.0%	0.6%	0.8%
Q4 2012	70.4%	20.9%	2.6%	3.2%	2.9%
Q4 2011	52.8%	23.0%	1.5%	8.1%	14.6%

Tabela 1: Quota de mercado das plataformas móveis [(11)]

O Android representa assim a plataforma móvel mais utilizado no mundo. Uma das razões é a quantidade de aparelhos que são lançados com o sistema Android instalado. Isto devido a fabricantes que fazem parte da OHA que o utilizam em seus aparelhos, de entre os quais pode-se destacar a Samsung, Motorola, LG, entre outros.

Outro ponto importante é que o Android pode ser encontrado em aparelhos de alto, médio e baixo custo.

1.2 Características básicas do Android

O sistema Android apresenta diversas características, dos quais as seguintes [(12)]:

- Maquina virtual Dalvik optimizada para dispositivos móveis, que permite as aplicações escritas em Java seja distribuído em formato binário (bytecode) e possam ser executados em qualquer dispositivo Android;
- O Android é adaptável para monitores de alta resolução, bibliotecas gráficas 2D e 3D;
- O Android tem suporte a vários idiomas;
- Suporte a vários meios de conectividade: WIFI, GSM, EDGE, 3G, 4G, Bluetooth;
- Apresenta um navegador Web baseado no Framework de código aberto denominado de Webkit;
- Uso da base de dados SQLite para armazenamento de dados;
- Suporte a diversos formatos de áudio, vídeo e imagens como: MPEG-4, H.264, MP3, AAC, PNG, JPG, GIF;
- **S**uporte a SMS e MMS para a troca de mensagens;
- Suporte a *hardware* adicional, isto é, o Android faz uso de câmaras de vídeo, GPS, acelerómetro, tela sensível de toque;

1.3 Versões do sistema operativo Android

Desde que foi lançado em 2007, o sistema operativo Android teve várias versões novas, cada uma corrigindo erros das versões anteriores e com novas funcionalidades. Curiosamente a partir de Abril de 2009, cada versão lançada recebe o nome de uma sobremesa (em Inglês) e seguem uma ordem alfabética [13].

Actualmente o Android dispõe das seguintes versões:

- Versão 1.0;
- Versão 1.1;
- Versão 1.5: Cupcake;
- Versão 1.6: Donut;
- Versão 2.0 - 2.1: Eclair;
- Versão 2.2: FroYo;
- Versão 2.3: Gingerbread;
- Versão 3.0/3.2: Honeycomb;
- Versão 4.0: Ice Cream Sandwich;
- Versão 4.1/4.2/4.3: Jelly Bean;
- Versão 4.4: KitKat;
- Versão 5.0: Lollipop.

1.3.1 Versão 1.0

A versão 1.0 do Android é a primeira versão comercial do sistema a ser lançada, e foi anunciada em 23 de Setembro de 2008. Esta versão dispõe das seguintes características:

- Suporte a câmera fotográfica;
- Navegador Web;
- Reprodução de vídeos através do Youtube;
- Aplicação Android Market para fazer *download* e actualizar as aplicações.

A figura 1 exibe o logótipo da versão 1.0 [14].

Figura 1: Logótipo do Android da versão 1.0 [14]

1.3.2 Versão 1.1

A versão 1.1 foi a primeira actualização do sistema operativo Android, lançada em 9 de Fevereiro de 2009. Esta versão foi elaborada para corrigir as falhas e melhorar as características da versão anterior [14]. A figura 2 apresenta o logótipo desta versão.

Figura 2: Logótipo do Android da versão 1.1 [14]

1.3.3 Versão 1.5 - Cupcake

A versão 1.5 do sistema operativo Android foi lançada em 30 de Abril de 2009 e foi a primeira a ser apelidada com o nome de uma sobremesa (em Inglês), o que sucedeu em todas as versões seguintes. A versão 1.5 sofreu grandes alterações, dos quais são apresentadas as seguintes [14]

- Possibilidade de gravar vídeos;
- Sistema de auto-rotação;
- Correcção automática nos textos;

- Suporte a tecnologia Bluetooth;
- Teclado virtual;
- *Upload* de vídeos e fotografias.

A figura 3 mostra o logótipo desta versão do sistema Android.

Figura 3: Logótipo do Android da versão Cupcake [14]

1.3.4 Versão 1.6 - Donut

A versão 1.6 do sistema operativo Android foi lançada em 15 de Setembro de 2009. Apelidada de Donut esta versão inclui actualizações e novos recursos, nomeadamente [14]:

- Suporte a telas de maior resolução;
- Inclusão de uma caixa de pesquisa rápida na Interface inicial;
- *Text-to-speech*: sistema de interpretação de palavras escritas através de voz;

O logótipo do Android da versão Donut é exibido na figura 4.

Figura 4: Logótipo do Android da versão Donut [14]

1.3.5 Versão 2.0/ 2.1 - Eclair

Foi a primeira vez que duas versões do Android receberam o mesmo nome de sobremesa. A versão 2.0 do sistema operativo Android foi anunciada em 26 de Outubro de 2009 e actualizada para a versão 2.1 em 11 de Janeiro de 2010. Esta versão disponibilizou os seguintes actualizações [(14)]:

- Suporte ao Bluetooth 2.1;
- *Browser* com suporte HTML e zoom;
- Sistemas de sincronização de contactos de emails e redes sociais;
- Possibilidades de inclusão de varias contas de email no aparelho;
- Actualização da aplicação câmera fotográfica, com a implementação das opções de *flash*, zoom e efeitos nas fotografias.

Na figura 5 pode-se observar o logótipo do Android da versão Eclair.

Figura 5: Logótipo do Android da versão Eclair [(14)]

1.3.6 Versão 2.2 - Froyo

Em 20 de Maio de 2010, foi lançada a versão 2.2 do sistema Android e recebeu o apelido de Froyo, abreviatura de Frozen Yogurt. Esta versão foi marcada por várias novidades, nomeadamente [(14)]:

- Suporte a adobe *flash*;
- Possibilidade de utilizar o Android Market (agora denominado de Google Play) para actualizar automaticamente as aplicações;
- Implementação da função USB *Tethering*, função que permite compartilhar internet via USB;

- Implementação da função Wi-Fi *Hotspot*, que permite compartilhar internet via Wi-Fi;
- Maior desempenho em termos de velocidade de processamento;
- Possibilidade de instalar aplicações em cartões de memória removíveis;
- Troca de idiomas no teclado.

A figura 6 mostra o logótipo do Android da versão Froyo.

Figura 6: Logótipo do Android da versão Froyo [(14)]

1.3.7 Versão 2.3 - Gingerbread

A versão 2.3 do sistema Android foi anunciada em 6 de Dezembro de 2010 e foi denominado de Gingerbread. Nesta versão foram implementados e actualizados os seguintes recursos [(14)]:

- Introdução da tecnologia NFC para a transmissão de dados;
- Suporte a resolução HD;
- Renovação da interface;
- Suporte a multi-câmeras em um mesmo aparelho;
- Uso dos serviços baseados em voip, nomeadamente a aplicação Google Talk para chamadas de voz e vídeo;
- Suporte a sensores como barómetro e giroscópio;
- Melhoria na autonomia da bateria;

A figura 7 apresenta o logótipo do Android da versão Gingerbread.

Figura 7: Logótipo do Android da versão Gingerbread [(14)]

1.3.8 Versão 3.0/ 3.1/3.2 - Honeycomb

A versão 3.0 do sistema Android foi lançada em 22 de Fevereiro de 2011, constituindo a primeira versão do Android disponibilizada para os *tablets*. A interface da versão Honeycomb foi optimizada para dispositivos com resolução de telas maiores. A actualização desta versão incluiu as seguintes funcionalidades [(14)]:

- Suporte para teclado em dispositivos com telas grandes;
- Interface gráfica nova;
- Suporte a processadores com múltiplos núcleos;
- Suporte da tecnologia Tethering através do Bluetooth;

Na figura 8 pode-se observar o logótipo do Android da versão Honeycomb.

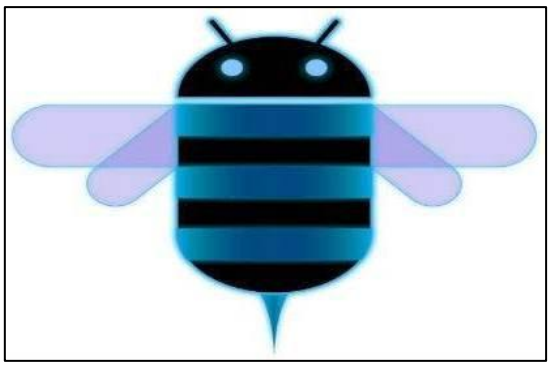


Figura 8: Logótipo do Android da versão Honeycomb [(14)]

1.3.9 Versão 4.0 - Ice Cream Sandwich

O Android versão 4.0 foi lançado publicamente em 19 de Outubro 2011 e trouxe mudanças no visual e novas funcionalidades. A versão denominada Ice Cream Sandwich incluiu as seguintes características [14]:

- Desbloqueio do *smartphone* através do reconhecimento facial;
- Câmera com modo panorama e possibilidade de aumentar o *zoom* enquanto faz uma gravação;
- Suporte a Wi-Fi Direct;
- Suporte a pastas;
- Possibilidade de tirar *screenshot*, possibilidade de capturar imagens do *display*;

Na figura 9 apresenta-se o logótipo do Android da versão Ice Cream Sandwich.

Figura 9: Logótipo do Android da versão Ice Cream Sandwich [14]

1.3.10 Versão 4.1/4.2/4.3- Jelly Bean

A versão 4.1 do Android foi lançada em 27 de Junho de 2012, denominada de Jelly Bean. As edições as 4.2 e 4.3 foram lançadas em 26 de Outubro de 2012 e 24 de Julho de 2013 respectivamente. Esta versão trouxe uma interface mais renovada, incluindo os seguintes recursos [14]:

- *Widgets* redimensionáveis;
- Suporte a saída de *áudio* via USB;
- Suporte a vários utilizadores no sistema operativo;
- Suporte a Open GL ES 3.0, uma especificação de gráficos 3D acelerada para dispositivos portáteis;

Na figura 10 apresenta-se o logótipo do Android da versão Jelly Bean.

Figura 10: Logótipo do Android da versão Jelly Bean [14]

1.3.11 Versão 4.4 - Kit Kat

A versão 4.4 do Android foi anunciada publicamente a 31 de Outubro de 2013. A versão 4.4 do Android apresentou novas funcionalidades e alterações do *design*. Das funcionalidades apresentadas pela referida versão, são apresentadas as seguintes [15]:

- Identificação de chamadas mais inteligentes;
- Suporte a impressão via Wi-Fi;
- Desempenho multitarefa rápido;
- Possibilidade de ter todas as mensagens num único local;
- Suporte para Bluetooth MAP, permite que o dispositivo possa conectar com carros que suportam a tecnologia do Bluetooth;

Na figura 11 se mostra o logótipo do Android da versão Kit Kat.

Figura 11: Logótipo do Android da versão Kit Kat [16]

1.3.12 Versão 5.0 - Lollipop

O Android 5.0 foi lançado publicamente em 3 de Novembro de 2014 e trata até o momento a versão mais recente disponibilizada pelo Google. Esta versão para além de ser lançada para *smartphones* e *tablets*, também se estende a relógios, a televisão e até mesmo carros. A versão 5.0 do Android trouxe uma nova interface, incluindo os seguintes recursos [(17)]:

- Suporte a múltiplos utilizadores, isto é, possibilidade de fazer *login* em outro dispositivo Android com o Lollipop, podendo efectuar uma ligação, ver SMS, fotografias, entre outros;
- Maior autonomia da bateria;
- Possibilidade de ver e responder mensagens directamente de sua tela de bloqueio;
- Possibilidade de priorizar aplicações, permite configurar uma lista de aplicações que podem receber notificações no modo " Não perturbe";
- Possibilidade de bloquear a tela em algumas aplicações;
- Maior eficiência na conectividade com a internet permitindo alternar entre redes sem interrupção da conectividade.
- O suporte a Open GL ES 3.1 Khronos para aplicações 2D de mais alto desempenho e recursos gráficos 3D.

Na figura 12 mostra-se o logótipo do Android da versão Lollipop.

Figura 12: Logótipo do Android da versão Lollipop [(18)]

1.3.13 Distribuição por versão

A tabela 2 e o gráfico 1 apresentam a distribuição de cada versão para os dispositivos que acessaram ao Google Play. Os dados foram recolhidos durante um período de 7 dias que terminou no dia 2 de Fevereiro de 2015. As versões com distribuição inferior a 0.1% não são apresentadas.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.4%
4.1.x	Jelly Bean	16	18.4%
4.2.x		17	19.8%
4.3		18	6.3%
4.4	KitKat	19	39.7%
5.0	Lollipop	21	1.6%

Tabela 2: Distribuição das versões do Android [19]

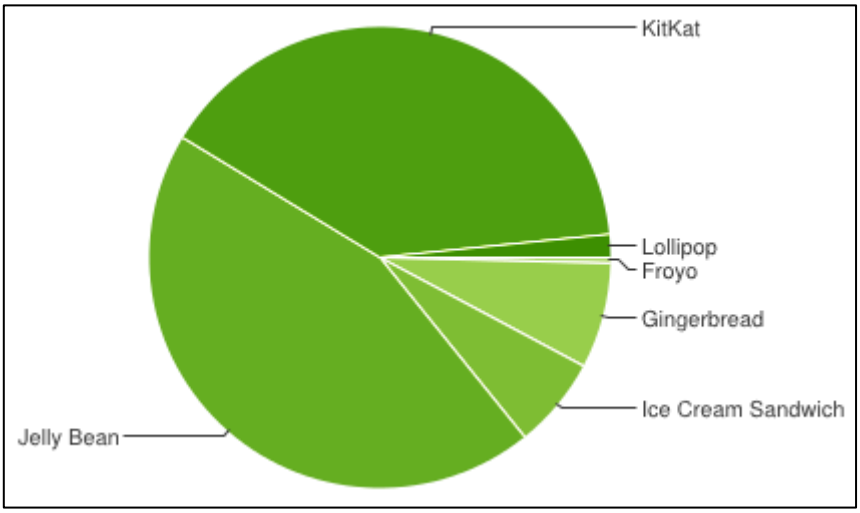


Gráfico 1: Distribuição das versões do Android [19]

1.4 A Arquitectura do Android

O sistema operacional Android é visto como uma pilha de *software*. A sua arquitectura é composta basicamente por 5 camadas, onde cada camada da pilha agrupa vários programas que suportam funções específicas do sistema [20]. Dos quais são:

- Kernel Linux;
- Bibliotecas;
- Android Runtime;
- Framework de Aplicação;
- Aplicação.

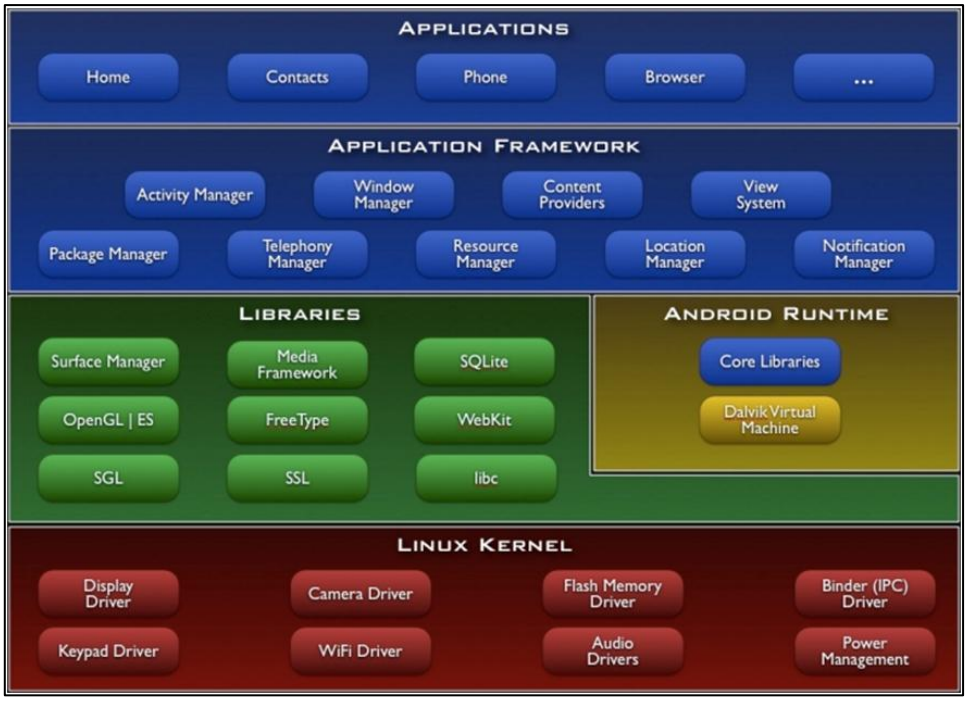


Figura 13: Arquitectura do Android [21]

1.4.1 Kernel do Linux

O núcleo do sistema operativo Android é derivado do Kernel 2.6 do Linux, e portanto, aproveita diversas características desta plataforma. A camada Kernel do Linux é considerada a base da pilha e responsável para os serviços centrais do sistema Android, tais como, as configurações de segurança, gestão de memória, gestão de processos e vários *drivers* de *hardware* [20].

Figura 14: Kernel do Linux [21]

1.4.2 Bibliotecas

Nesta camada estão as API´s desenvolvidas em C/C++ utilizadas por diversos componentes do sistema. Esta camada compreende um conjunto de instruções que permitem ao dispositivo manipular diferentes tipos de dados. A camada Bibliotecas permite trabalhar com arquivos de mídia, visualização 2D e 3D e ainda, possui funções para navegadores, aceleração do *hardware*, assim como para o acesso à base de dados SQLite [20].

Figura 15: Bibliotecas [21]

1.4.3 Android Runtime

A camada Android Runtime localiza-se no mesmo nível da camada Bibliotecas e integra a **Dalvik Virtual Machine**, que é uma máquina virtual criada especificamente para o sistema Android fornecendo condições para permitir a execução das aplicações. Esta camada também é composta pelo conjunto de bibliotecas denominadas de **Core Libraries**, que disponibilizam a API Java necessária para a escrita do código de programação das aplicações. A camada Android Runtime foi desenvolvida exclusivamente para o sistema Android, para permitir rodar programas com recurso a ambientes com pouco capacidade de memória, energia e níveis de processamento [20].

Figura 16: Android Runtime [21]

1.4.4 Framework de Aplicação

A camada Framework de Aplicação abrange um conjunto de API´s que fazem a gestão das funções básicas do telefone quando uma aplicação é executada sobre a plataforma Android. Esses API´s incluem os gerenciadores de localização, de actividades, de telefonia [20].

Figura 17: Framework de Aplicação [21]

1.4.5 Aplicação

A camada de Aplicação representa o topo da arquitectura do Android, composta por aplicações-chaves relacionadas com os serviços de email, navegadores, calendário, ente outros e também aplicações desenvolvidas por terceiros. Um utilizador comum interage com esta camada para acessar as funções básicas, enquanto as camadas mais baixas são acessadas por desenvolvedores, programadores [20].

Figura 18: Aplicação [21]

1.5 Componentes da aplicação Android

Uma aplicação Android é constituída por um conjunto de componentes que são formados por blocos de códigos essenciais. Cada componente possui uma função específica dentro da aplicação, porem não é obrigatório fazer uso de todos os componentes em todas as aplicações.

Uma aplicação Android é essencialmente composta pelos seguintes 4 componentes básicos [(22)]:

- Activity;
- Service;
- Content Provider;
- Broadcast Receiver.

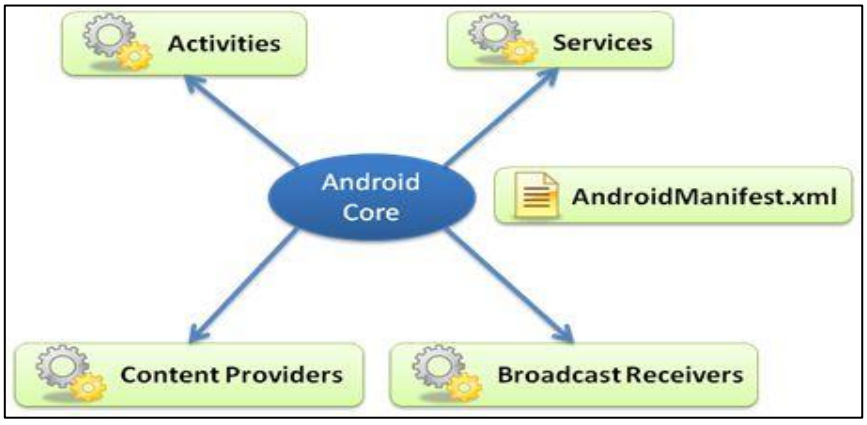


Figura 19: Componentes de uma aplicação Android [(23)]

1.5.1 Activity

A Activity é a componente de aplicação mais comum e representa uma tela com o qual os utilizadores podem interagir. Através de uma Activity pode-se realizar acções tais como enviar um email, fazer uma chamada, tirar uma fotografia.

Uma aplicação geralmente consiste em múltiplas activities interligadas entre si, em que uma Activity é especificada como principal. A Activity principal é apresentada ao utilizador quando a aplicação iniciar pela primeira vez.

Uma Activity é uma Subclasse da superclasse android.app.Activity [(24)].

1.5.2 Service

O Service é a componente de aplicação responsável para executar operações em segundo plano por um tempo indeterminado e não fornece uma interface ao utilizador. Um componente, como uma Activity, pode iniciar o Service e deixa-lo executando ou conectar-se nele para interagir com o mesmo. Como exemplo um Service pode reproduzir uma música, manipular transacções com a rede em segundo plano, se o utilizador estiver em uma aplicação diferente.

A componente Service é implementado da superclasse android.app.Service [(25)].

1.5.3 Content Provider

O Content Provider é o componente utilizado pela plataforma Android para acessar dados de um aplicativo Android, para compartilhar dados com outras aplicações que se encontram em execução no dispositivo. Esta componente permite centralizar os dados em um local único, de modo que aplicações diferentes possam acessar esses dados de acordo com as necessidades [(26).]

1.5.4 Broadcast Receiver

O Broadcast Receiver é um componente que funciona como um receptor de ocorrências de eventos, que podem ser próprios do sistema ou das aplicações e reage transmitindo informações. Este componente não possui uma interface visual, mas pode criar notificações na barra de *status* para avisar ao utilizador a ocorrência de um determinado evento [(22)].

Em adição a estes componentes também merecem destaque na arquitectura Android devido á sua importância, os seguintes ficheiros e classes:

- **AndroidManifest.xml**: O AndroidManifest.xml é um arquivo de existência obrigatória e única que cada aplicação dispõe. O arquivo AndroidManifest.xml contém informações gerais para executar uma aplicação, como, todos componentes da aplicação utilizados, nome do pacote. Neste arquivo também se definem as informações das permissões da aplicação Android [(27)];

- **Intent**: O Intent é objecto da classe android.content.Intent, e corresponde a uma descrição abstracta de uma operação a ser realizada. Uma Intent permite a interacção entre componentes da mesma aplicação, bem como entre componentes de outra aplicação. Este recurso é utilizado para iniciar uma Activity, um Service ou mesmo para entregar uma transmissão [28].

1.6 Interface de utilizador da aplicação Android

Todos os componentes de interface de utilizador são construídos usando a classe **view** [29]. Esta classe oferece as subclasses **View** que são componentes de interface e **ViewGroup** que são os layouts de interface.

1.6.1 Componentes de interface

Os componentes de interface são elementos gráficos que compõem as interfaces de uma aplicação. Estes elementos são subclasses da classe android.view e são denominados de widget [30]. A seguir serão apresentadas os widgets que foram utilizados no desenvolvimento da aplicação SOS Mindelo:

- **Button**: o widget Button exibe um botão que pode ser pressionado pelo utilizador para executar uma acção;
- **TextView**: o widget TextView exibe um texto ao utilizador onde é possível mostrar alguma informação, mensagem, etc;
- **ImageButton:** o widget ImageButton exibe um botão com uma imagem que pode ser pressionado para realizar uma acção;
- **ImageView:** o widget ImageView é responsável para apresentar imagens nas interfaces de utilizador;
- **ListView**: o widget ListView exibe itens que podem ser seleccionados em uma lista de rolagem vertical.

1.6.2 Layouts de interface

Os Layouts de Interface são componentes da subclasse ViewGroup que definem a estrutura visual para uma interface de utilizador. Os layouts são utilizados para orientar o posicionamento dos widgets na interface do utilizador. A seguir serão apresentados os layouts utilizados para o desenvolvimento da aplicação SOS Mindelo:

- **LinearLayout**: O LinearLayout organiza os componentes de interface na tela em uma única direcção, que pode ser na horizontal ou na vertical;
- **RelativeLayout**: O RelativeLayout organiza os componentes de interface na tela em posições relativas. Este layout possibilita alinhar dois elementos no centro, na esquerda ou direita por cima ou por baixo uma da outra.

1.7 Ciclo de vida de uma aplicação Android

Todo o componente do sistema que interagir de qualquer modo com o utilizador deverá ter pelo menos uma ou mais activities. No Android apenas uma Activity pode estar activa, por isso existe a Activity Stack, ou Pilha de Actividade para efectuar o controlo de todas as activities. As activities vão fazendo parte da pilha à medida que vão sendo executados.

Quando uma aplicação se inicia, a Activity principal entra em execução e é colocada no topo da pilha. Ao iniciar uma nova Activity, a Activity principal perde foco e fica em baixo da Activity nova que é colocado no topo da pilha. Sendo assim sempre que uma nova Activity inicia, é colocada no topo da pilha e a Activity que estava activa fica logo abaixo da Activity nova.

O ciclo de vida de uma aplicação Android compreende em um conjunto de acções que são tomadas quando uma ou mais aplicações estão em execução. A figura 8 ilustra o fluxograma do ciclo de vida de uma aplicação Android [(31)]:

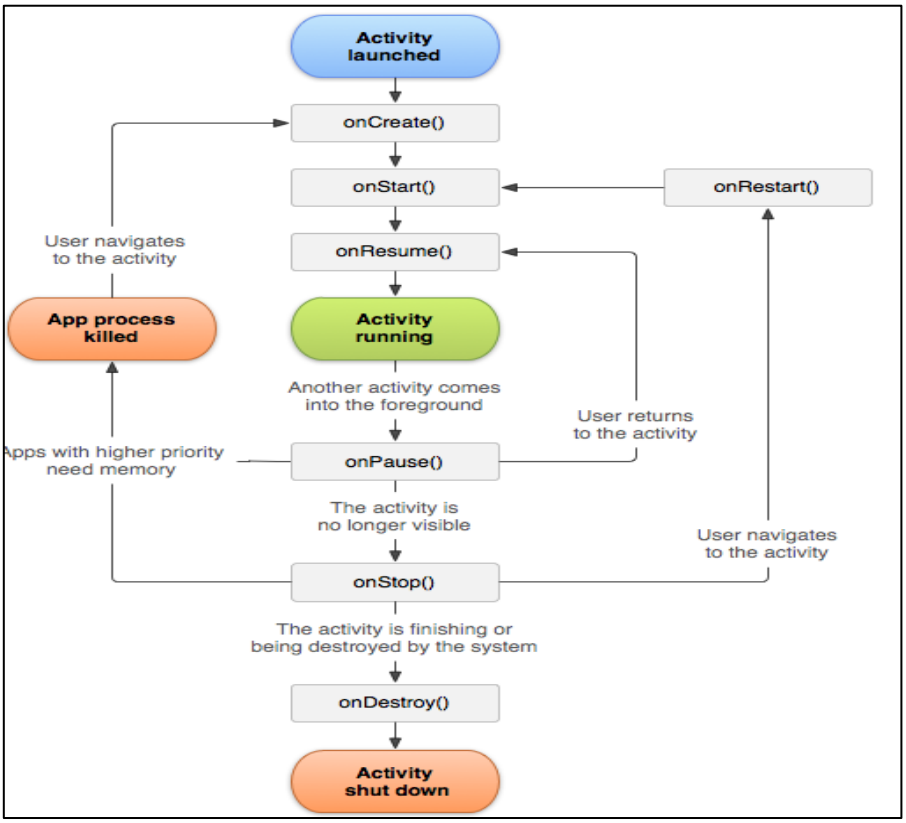


Figura 20: Ciclo de vida de uma aplicação Android [31]

Em geral, o ciclo de vida de uma aplicação Android compreende entre os seguintes métodos:

- **onCreate() –** é o primeiro método a ser executado quando uma Activity se inicia pela primeira vez, é de uso obrigatório e invocado apenas uma única vez. Este método é também utilizado para carregar os layouts XML, carregar os dados a serem exibidos, inicializar variáveis;
- **onStart() –** é chamado logo após o método onCreate(), e também quando uma Activity que estava em segundo plano volta a ter foco. Este método indica que a Activity está prestes a ser exibida na tela**;**

- **onResume() –** é chamado quando a Activity vai iniciar a interacção com o utilizador, é chamado após o método onStart(). Neste método a Activity encontra-se no topo da pilha**;**

- **onPause() –** este método é chamado quando a Activity que se encontra no topo da pilha está a perder o foco para outra Activity. Neste método a Activity actual é suspensa, ficando à espera de uma possível futura retomada para o topo da pilha**;**
- **onStop() –** este método é chamado quando a Activity em questão não estiver mais visível, tendo pois perdido o primeiro plano. A Activity em causa está sendo destruída porque outra Activity, seja existente ou nova, está indo para o primeiro plano e cobrindo-o**;**
- **onDestroy() –** ultimo método a ser chamado. Depois deste método a Activity em causa passa a ser considerada morta, e é removida por completa da pilha, dando o encerramento completo do processo;
- **onRestart() –** é chamado imediatamente antes do método onStart(), quando uma Activity volta a estar no topo da pilha depois de estar em segundo plano.

1.8 Descrição das ferramentas utilizadas de desenvolvimento para Android

Para o desenvolvimento de aplicativos para o sistema operativo Android encontram-se disponíveis ferramentas que podem ser utilizadas de forma gratuita em ambientes Windows, Mac OS e Linux. Para o desenvolvimento deste projecto foram utilizadas as seguintes ferramentas:

- **Android SDK:** O Android SDK inclui um conjunto de ferramentas necessárias para a construção de aplicações Android. O Android SDK tem as seguintes ferramentas: Android SDK Manager, AVD Manager, emulador, depurador, documentação, entre outros [(32)].
- **IDE Eclipse:** O IDE Eclipse é uma ferramenta composta por elementos utilizados no desenvolvimento de *software*, tais como, editor de código, depurador, entre outros. O IDE Eclipse é mais frequentemente utilizado para o desenvolvimento com a linguagem Java, mas pode ser utilizado para desenvolver em outras linguagens mediante uso de *plugin*. Por exemplo, o *plugin* PDT é utilizado no eclipse para permitir o desenvolvimento com uso

da linguagem PHP. O IDE Eclipse também possibilita fazer a integração com o Android SDK [33];

- **ADT Plugin:** O ADT Plugin é uma extensão desenvolvido para o Eclipse com um conjunto de ferramentas, utilizadas em projectos para o Android. O ADT Plugin oferece uma interface gráfica para construir interfaces de aplicações. Este *plugin* utiliza as bibliotecas do SDK, fazendo a adaptação entre o Android SDK e a IDE Eclipse [34];
- **JDK:** O JDK é composto por um conjunto de ferramentas básicas, tais como javadoc (gerador automático de documentação), javac (compilador), jdb (Java debugger), entre outros, para o desenvolvimento de aplicações utilizando a linguagem Java [35];

2 Desenvolvimento da aplicação

2.1 Especificação de Requisitos

Os requisitos desta aplicação encontram-se agrupados nas seguintes duas grandes categorias:

- Requisitos funcionais;
- Requisitos não funcionais.

2.1.1 Requisitos Funcionais

Os requisitos funcionais definem as funcionalidades do sistema que funcionam como respostas do sistema às solicitações dos utilizadores. Os requisitos funcionais da aplicação são enumerados a seguir:

- Mostrar farmácias mais próximas;
- Mostrar Hospitais mais próximos;
- Mostrar Centros Saúde mais próximas;
- Mostrar Clinicas mais próximas;
- Exibir informações acerca das farmácias de serviço da cidade do Mindelo;
- Realizar chamadas para os locais presentes na aplicação;
- Efectuar chamadas para números de emergências;
- Enviar email para as entidades de emergências;
- Determinar a localização do utilizador;
- Localizar os lugares presentes na aplicação no mapa;
- Ver uma rota de destino entre a posição do dispositivo e do lugar pretendido
- Mostrar a distância do utilizador para o local escolhido.

2.1.2 Requisitos não Funcionais

Os requisitos não funcionais definem as características, restrições necessárias para a implementação das funcionalidades mas que não são respostas directas aos estímulos dos utilizadores. Este trabalho apresenta os seguintes requisitos funcionais:

- Disponibilidade de conexão com á internet para disponibilizar informações ao utilizador;
- Uso do GPS para determinar as coordenadas geográficas do dispositivo;
- Uso do API Google Maps para determinar e visualizar rotas;
- Reconhecer a opção seleccionada no menu;
- Hospedagem da Base de Dados *online* para disponibilizar informações ao utilizador;
- Requer a versão 2.2 ou superior do sistema operativo Android para executar a aplicação;

2.2 Modelação

A modelação do sistema faz parte do processo de desenvolvimento de *software* e consiste em construir modelos para melhor explicar as características ou o comportamento de um software. Utilizou-se a modelação UML para ilustrar as funcionalidades protótipo desenvolvido.

A UML é uma linguagem padrão de modelação que permite representar um sistema. A UML utiliza a notação de diagramas para a especificação, visualização, construção e documentação de *softwares* [(36)].

2.2.1 Diagrama de Casos de Uso

A figura 21 apresenta o diagrama de casos de uso onde se especificam os relacionamentos entre as diferentes partes da aplicação.

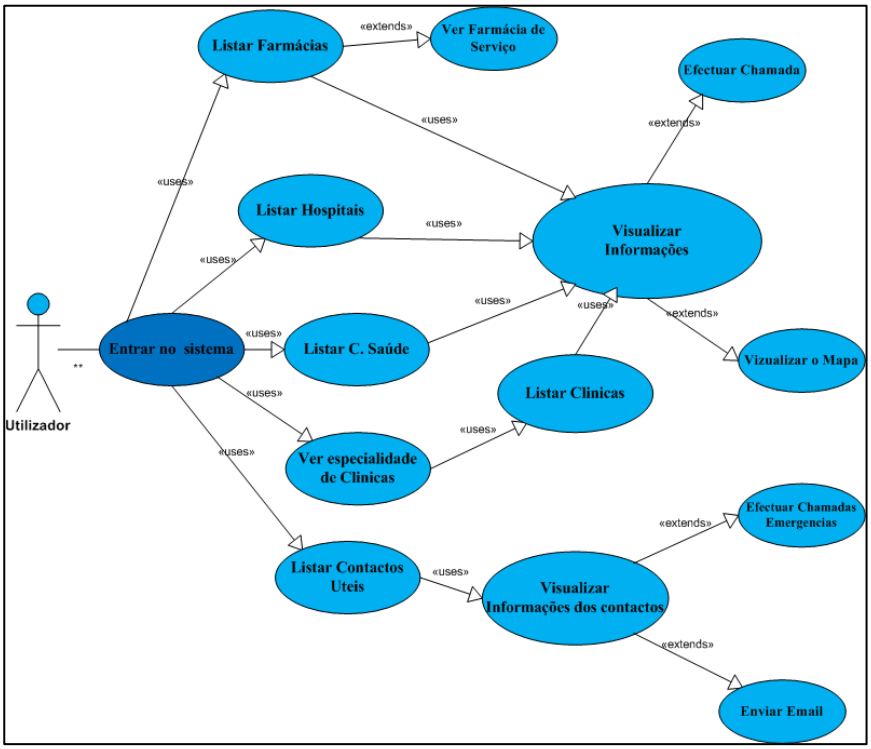


Figura 21: Diagrama de Casos de Uso

Os casos de uso apresentados na figura 21 são descritos a seguir:

- Entrar no sistema;
- Listar farmácias;
- Mostrar especialidades de clinicas;
- Listar clinicas;
- Listar hospital;
- Listar centros saúde;
- Listar contactos úteis;
- Ver farmácias de serviço;
- Visualizar Informações;
- Efectuar chamadas;
- Visualizar o mapa;
- Enviar email;
- Visualizar Informações dos contactos;
- Efectuar chama de emergências;

Caso de Uso Entrar no sistema

- **Finalidade**: apresentar a interface inicial da aplicação;
- **Sequência típica de eventos:**
  1. O utilizador inicia a aplicação, seleccionando o ícone da aplicação, disponível na área de aplicações do dispositivo;
  2. A aplicação apresenta ao utilizador a interface inicial da aplicação;
  3. O utilizador visualiza a interface inicial com todas as opções disponíveis, podendo seleccionar uma delas.

Caso de Uso Listar farmácias

- **Finalidade:** listar todas as farmácias existentes na cidade do Mindelo.
- **Sequência típica de eventos:**
  1. O utilizador executa o caso de uso Entrar no sistema;
  2. O Utilizador selecciona a opção listar farmácias;
  3. A aplicação obtém da base de dados remota informações de todas as farmácias;
  4. A aplicação disponibiliza a lista de farmácias, o utilizador visualiza os itens da lista, podendo seleccionar um deles.

Caso de Uso Ver farmácia de serviço

- **Finalidade:** listar todas as farmácias e as respectivas datas em que estão de serviço.
- **Sequência típica de eventos:**
  1. O utilizador executa o Caso de Uso Listar farmácias;
  2. O utilizador selecciona a opção ver Farmácia de serviço;
  3. A aplicação disponibiliza ao utilizador as farmácias e as respectivas datas que estão de serviço.

Caso de Uso Mostrar especialidade de clinicas

- **Finalidade:** listar as especialidades de clinicas existentes na cidade do Mindelo.
- **Sequência típica de eventos:**
  1. O utilizador executa o caso de uso Entrar no sistema;
  2. O Utilizador selecciona a opção Mostrar especialidade de clinicas;
  3. O utilizador visualiza a interface com as opções especialidades de clinicas, podendo seleccionar uma das opções.

Caso de Uso Listar clinicas

- **Finalidade:** listar as clinicas existentes na cidade do Mindelo, de acordo com a especialidade escolhida.
- **Sequência típica de eventos:**
  1. O utilizador executa o caso de uso Entrar no sistema e Mostrar especialidade de clinicas;
  2. A aplicação obtém da base de dados remota informações de todas as clinicas, conforme a especialidade escolhida;
  3. A aplicação disponibiliza a lista de clinicas, o utilizador visualiza os itens da lista, podendo seleccionar um dos itens.

Caso de Uso Listar hospital

- **Finalidade:** mostrar o hospital da cidade do Mindelo.
- **Sequência típica de eventos:**
  1. O utilizador executa o caso de uso Entrar no sistema;
  2. O Utilizador selecciona a opção listar hospital;
  3. A aplicação obtém da base de dados remota informações do hospital;
  4. A aplicação apresenta o hospital, o utilizador visualiza o item da lista, podendo seleccionar o item.

Caso de Uso Listar centros saúde

- **Finalidade:** listar os centros de saúde da cidade do Mindelo.
- **Sequência típica de eventos:**
  1. O utilizador executa o caso de uso Entrar no sistema;
  2. O Utilizador selecciona a opção listar centros saúde;
  3. A aplicação obtém da base de dados remota informações de todos os centros saúde;
  4. A aplicação disponibiliza a lista de centros saúde, o utilizador visualiza os itens da lista, podendo seleccionar um deles.

Caso de Uso Listar contactos úteis

- **Finalidade:** listar os contactos úteis da cidade do Mindelo.
- **Sequência típica de eventos:**
  1. O utilizador executa o caso de uso Entrar no sistema;
  2. O Utilizador selecciona a opção listar os contactos úteis;
  3. A aplicação obtém da base de dados remota os contactos úteis;
  4. A aplicação disponibiliza a lista dos contactos úteis, o utilizador visualiza os itens da lista, podendo seleccionar um deles.

Caso de Uso Visualizar informações

- **Finalidade:** ver com mais detalhes informações de um dos lugares presente na aplicação.
- **Sequência típica de eventos:**
  1. O utilizador executa um dos seguintes o caso de uso: Listar farmácias, listar clinicas, listar hospital ou listar centros saúde;
  2. A aplicação disponibiliza ao utilizador os dados do lugar escolhido, com as opções de chamar e visualizar o lugar no mapa, juntamente com a sua rota de destino;
  3. O utilizador visualiza com mais detalhes as informações do lugar escolhido e as opções mencionadas, podendo seleccionar uma delas.

Caso de Uso Efectuar chamadas

- **Finalidade:** efectuar chamada a um dos lugares presentes na aplicação, de acordo com a escolha do utilizador.
- **Sequência típica de eventos:**
  4. O utilizador executa o Caso de Uso Visualizar informações;
  5. O utilizador selecciona a opção chamar;
  6. A aplicação obtém o contacto telefónico do lugar escolhido e efectua a ligação.

Caso de Uso Visualizar o mapa

- **Finalidade:** visualizar a localização do utilizador e o lugar no mapa, juntamente com a sua rota de destino.
- **Sequência típica de eventos:**
  1. O utilizador executa o Caso de Uso Visualizar informações;
  2. O utilizador selecciona a opção ver mapa;
  3. A aplicação obtém as coordenadas do utilizador e do lugar escolhido, localizando-os no mapa e apresentando uma rota de destino.

Caso de Uso Visualizar Informações dos contactos

- **Finalidade:** ver com mais detalhes informações dos contactos úteis existentes na cidade do Mindelo.
- **Sequência típica de eventos:**
  1. O utilizador executa o caso de uso Listar contactos úteis;
  2. A aplicação disponibiliza ao utilizador os dados do contacto escolhido, juntamente com opções de efectuar chamadas de emergências e enviar email;
  3. O utilizador visualiza com mais detalhes as informações do contacto escolhido e as opções mencionadas, podendo seleccionar uma delas.

Caso de Uso Efectuar chamadas de emergências

- **Finalidade:** efectuar chamada de emergências para os contactos úteis presentes na aplicação, de acordo com a escolha do utilizador.
- **Sequência típica de eventos:**
  1. O utilizador executa o Caso de Uso Visualizar informações dos contactos;
  2. O utilizador selecciona a opção chamar;
  3. A aplicação obtém o contacto telefónico do lugar escolhido e efectua a ligação.

Caso de Uso enviar Email

- **Finalidade:** efectuar o envio de email para os contactos úteis presentes na aplicação, de acordo com a escolha do utilizador.
- **Sequência típica de eventos:**
  1. O utilizador executa o Caso de Uso Visualizar informações dos contactos;
  2. O utilizador selecciona a opção enviar email;
  3. A aplicação obtém o email do lugar escolhido e efectua o envio do email.

2.2.2 Diagrama de Classes

A figura 22 mostra o diagrama de classes da aplicação onde se podem observar as classes utilizadas. Cada Activity é representada como uma classe, juntamente com os seus atributos e funções.

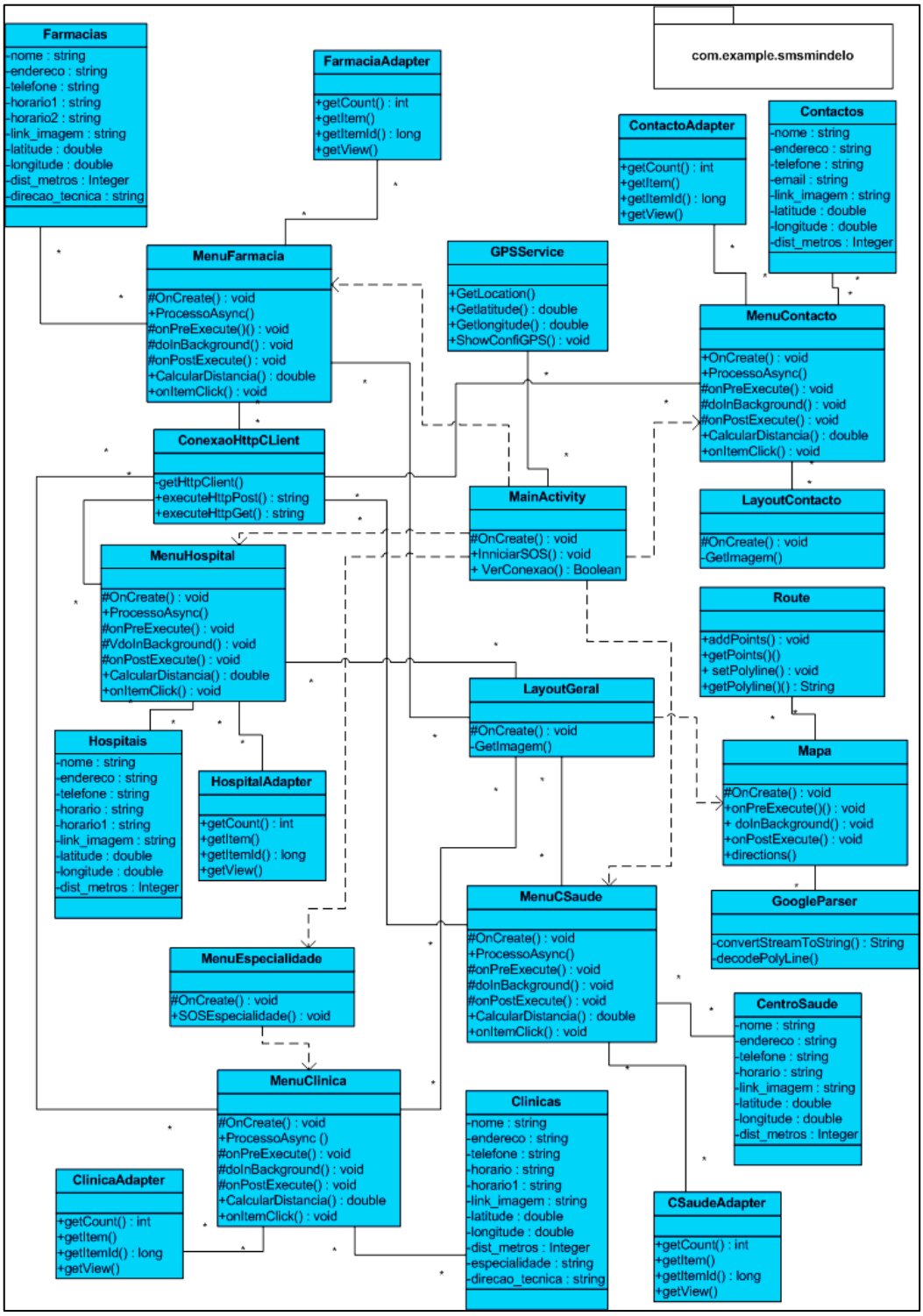


Figura 22: Diagrama de classes

No diagrama de classes, figura 22, observa-se a classe **MainActivity**, que é a classe principal da aplicação e representa a primeira classe que é chamada ao iniciar a aplicação. Esta classe estende a classe Activity, que apresenta a interface inicial ao utilizador e é composta pelos métodos:

- **onCreate**: O método onCreate recebe a referência da interface inicial, exibindo-a ao utilizador, verifica se existe a conexão com a internet e obtém as coordenadas geográficas de latitude e longitude;
- **VerConexao**: O método VerConexao verifica se o estado de conectividade da rede do dispositivo através da classe ConnectivityManager [37]. Na figura 23 mostra-se um extracto de código desta função;

```
// verifica o estado de conexao da internet
public boolean VerConexao(){
    ConnectivityManager connMgr = (ConnectivityManager) getSystemService(MainActivity.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected())
            return true;
        else
            return false;
}
```

Figura 23: Extracto de código que verifica o estado da conexão

As classes **MenuFarmacia**, **MenuClinica**, **MenuHospital**, **MenuCSaude** e **MenuContacto** são responsáveis para exibir informações das farmácias, clinicas, hospitais, centros de saúde e contactos úteis respectivamente. Estas classes também se estendem a classe Activity. Foi criada, dentro das classes acima referidas, uma classe denominado de ProcessoAsync que herde Async Task [38].

O Async Task é um mecanismo que encapsula tarefas pesadas, fornecendo métodos para modificar a interface de utilizador antes, durante e depois da execução dessas tarefas. Este mecanismo foi implementado usando os seguintes métodos:

- onPreExecute;
- doInBackground;
- onPostExecute;

O método **onPreExecute** é chamado antes da execução da tarefa principal, isto é, antes do método doInBackground. Este método geralmente é usado para exibir ao utilizador uma mensagem de que algo esta sendo carregado, através de um ProgressDialog.

A implementação deste método é apresentada na figura 24.

```
protected void onPreExecute() {

    super.onPreExecute();
    dialog = new ProgressDialog(context);
    dialog.setTitle("Aguarde");
    dialog.setMessage("Carregando os dados ...");
    dialog.show();
}
```

Figura 24: Método onPreExecute

O método **doInBackground** realiza a tarefa principal em segundo plano. Este método é usado para executar tarefas que podem levar algum tempo para se executarem e é chamado quando o método onPreExecute se finaliza.

Nestas classes o método **doInBackground** é responsável por converter os dados do formato JSON para String, calcular a distância entre utilizador e os lugares pretendidos, adicionar os dados no ArrayList e ordenar o ArrayList por ordem crescente da distância. A figura 25 apresenta a implementação deste método na classe MenuFarmacia, as outras classes seguem a mesma implementação, diferenciando apenas na URL.

```
protected Void doInBackground(String... params) {
    // TODO Auto-generated method stub
    String resposta;

    Intent info= getIntent();
        if (info != null) {
             latitude=Double.parseDouble(info.getStringExtra("latitude"));
             longitude=Double.parseDouble(info.getStringExtra("longitude"));
             Log.i("Log_tag", "longitude:"+longitude+"  latitude:"+latitude);
    }
    try{
        resposta=ConexaoHttpCLient.executeHttpGet("http://sosmindelo.esy.es/farmacias.php");
        // Armazena o resultado retornado pelo script PHP que resulta da consulta MySQL
        String result=resposta.toString();
        try{
            JSONArray jArray = new JSONArray(result);
            for(int i=0;i<jArray.length();i++){
                JSONObject json_data = jArray.getJSONObject(i);
                // calcula a distancia
                double distancia=calcularDistancia(json_data.getDouble("latitude"),latitude,json_data.getDouble("l
                Log.i("Log_tag", "nome:"+json_data.getString("nome")+"  distancia:"+distancia);

                // armazena no arrayList um objecto do tipo farmacia juntamente com a sua distancia
                farmacias.add(new Farmacias(json_data.getString("nome"),json_data.getString("endereco"), json_data
                        json_data.getString("link_imagem"), json_data.getDouble("latitude"),json_data.getDouble("l
                        json_data.getString("horario1"),json_data.getString("horario2"),json_data.getString("direc
            }
            Collections.sort(farmacias);
```

Figura 25: Método doInBackground

O método **onPostExecute** é executado após o término da execução do método doInBackground. Neste método deve-se dispensar o ProgressDialog e exibir informações processadas na tela. A figura 26 mostra a implementação deste método na classe MenuFarmacia.

```
protected void onPostExecute(Void result) {
      //Define o Adapter
      listView.setAdapter(adapterListView);
      //Cancela progressDialogo
      dialog.dismiss();
}
```

Figura 26: Método onPostExecute

As classes MenuFarmacia, MenuClinica, MenuHospital, MenuCSaude e MenuContacto são compostas pelos seguintes métodos:

- Método calcularDistancia;
- Método onItemClick.

O método **calcularDistancia** é o responsável pelo cálculo da distância em metros entre a posição do utilizador e os locais pretendidos no momento solicitado, com base nas coordenadas geográficas latitude e longitude. Para se determinar a distância entre os dois pontos geográficos utilizou-se o Teorema de Pitágoras [39].

Tendo a localização do utilizador como um ponto e o local pretendido como outro ponto, é possível traçar as projecções destes pontos sobre eixos coordenados e obter um triângulo rectângulo como mostra a figura 27. O eixo das ordenadas representa as longitudes e o eixo das abcissas representa as latitudes.

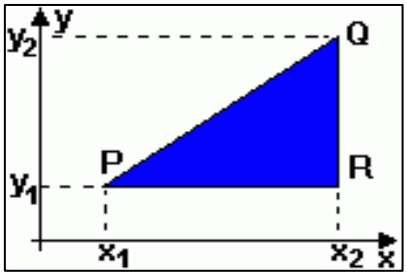


Figura 27: Triângulo de Pitágoras [40]

Para calcular a distância tendo em conta os dois pontos de coordenadas utilizou-se a seguinte fórmula apresentada na equação 1.

$$d(P,Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Equação 1: Cálculo do Teorema de Pitágoras [40]

Tendo em conta a fórmula apresentada, equação 1, a distância entre os dois pontos fica expressa em graus. Foi considerado que o valor de 6371 quilómetros para o raio da terra, que resulta num perímetro da terra de 40030000 metros com uso da equação 2. Para converter o resultado em graus para uma distância em metros utilizou-se a regra de 3 simples.

$$P = \pi * r * 2$$

Equação 2: Cálculo do perímetro da terra

A figura 28 apresenta a implementação do método calcularDistancia.

```
funcao  que permite calcular a distancia entre o dispositivo  e as farmacias
public double calcularDistancia(double lat,  double lat1, double lg, double lg1){
    double dist_metros=0.0;
    double DLO, DLA, d_ang;
    DLA=lat-lat1;
    DLO=lg-lg1;
    d_ang=Math.sqrt(Math.pow(DLO, 2)+ Math.pow(DLA, 2));
    dist_metros=(d_ang*40030000)/360;
    return dist_metros;
}//fim da funcao calcularDistancia()
```

Figura 28: Método calcular distancia

O Método onItemClick funciona quando um item da lista é seleccionado. Com este método obtém-se o Id da farmácia seleccionada juntamente com as respectivas informações, que serão enviadas como parâmetros para a Activity LayoutGeral. A Activity LayoutGeral é responsável por exibir informações detalhadas das farmácias. A figura 29 mostra a implementação do método onItemClick da classe MenuFarmacia.

```
// metodo que quando  um  item na listview e selecionado
public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
    //Pega o item que foi selecionado.
    Farmacias item = adapterListView.getItem(arg2);

    Intent farmacia= new Intent(this, TelaInformacoes.class);
    Bundle farm_params = new Bundle();
    farm_params.putString("nome", item.getNome());
    farm_params.putString("endereco",item.getEndereco());
    farm_params.putString("tel", item.getTelefone());
    farm_params.putString("link_imagem",item.getLinkImagem());
    farm_params.putString("horario1",item.getHorario1());
    farm_params.putString("horario2",item.getHorario2());
    farm_params.putString("direcao_tecnica",item.getDirecao_tecnica());
    farm_params.putString("latitude", item.getLatitude()+"");
    farm_params.putString("longitude", item.getLongitude()+"");
    farm_params.putString("latGPS", latitude+"");
    farm_params.putString("longGPS", longitude+"");
    farmacia.putExtras(farm_params);
    startActivity(farmacia);

}
```

Figura 29: Método onItemClick da Activity MenuFarmacia

As classes FarmaciaAdapter, ClinicaAdapter, HospitalAdapter, CSaudeAdapter e ContactoAdapter foram criadas para trabalhar com os dados a serem exibidos no ListView. O ListView foi implementado nas classes MenuFarmacia, MenuClinica, MenuHospital, MenuCSaude e MenuContacto. Estas Classes estendem a classe **BaseAdapter** [41] que implementa os seguintes métodos:

- **getCount()**: retorna o número de itens;
- **getItem()**: retorna o item de uma posição específica;
- **getItemId()**: retorna o Id de um item de uma posição específica;
- **getView()**: retorna a View com as informações posicionadas de acordo com o layout.

2.2.3 Diagrama de Sequências

Neste tópico são apresentados os diagramas de sequência que ilustram as interacções entre objectos de um cenário ao longo do tempo, realizadas através de operações ou métodos. Os diagramas de sequência foram construídos a partir do diagrama de Casos de Usos.

A figura 30 apresenta o diagrama de sequência do Caso de Uso **Listar farmácias**, onde se podem constatar as etapas que devem ser efectuadas no sistema por esta funcionalidade**.**

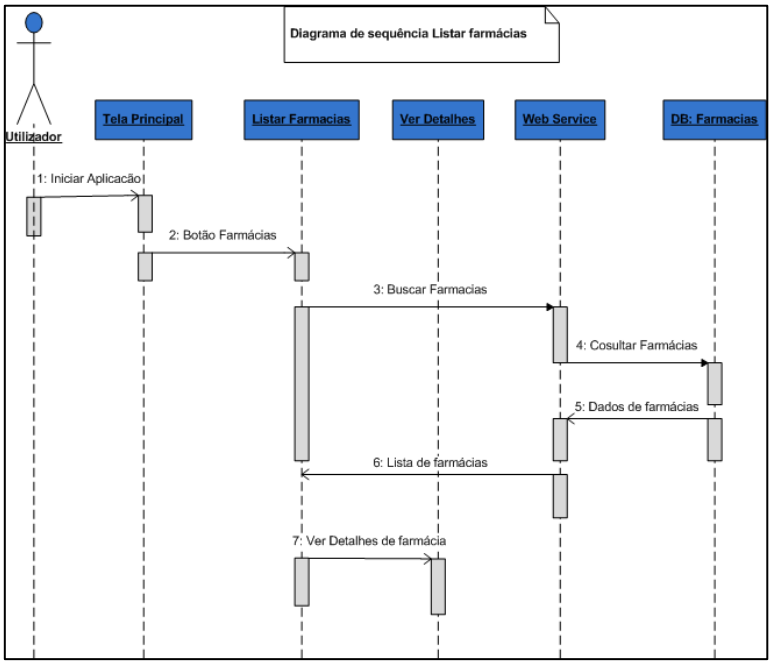


Figura 30: Diagrama de sequência ver farmácia

A figura 31 apresenta o diagrama de sequência do Caso de Uso **Listar clinicas**.

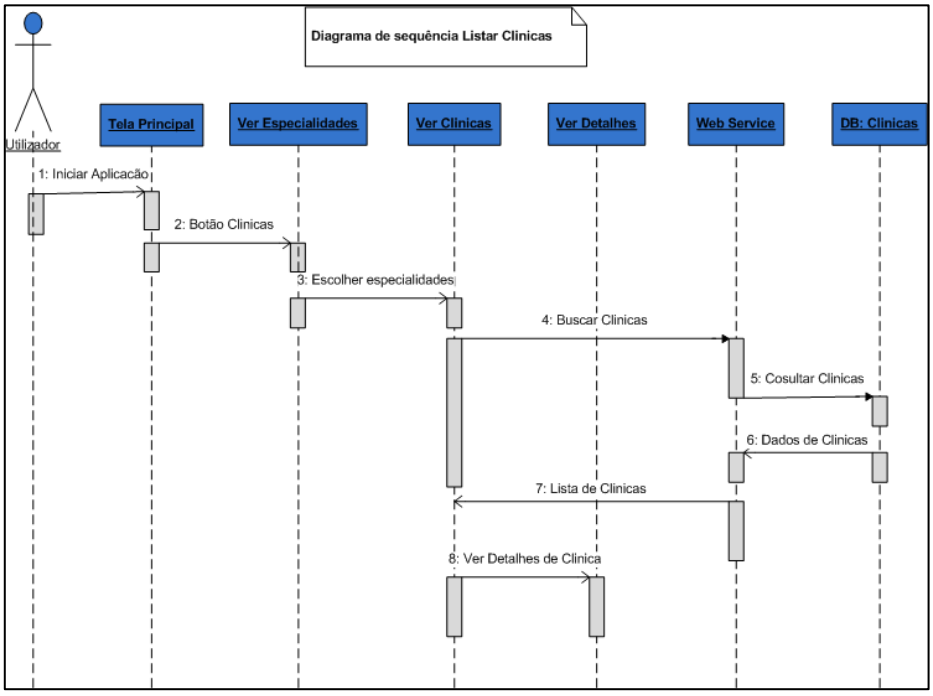


Figura 31: Diagrama de sequência ver clinicas

2.3 Implementação

2.3.1 Aplicação SOS Mindelo

A aplicação SOS Mindelo é uma aplicação para dispositivos com o sistema operativo Android, capaz de disponibilizar informações referentes a farmácias, hospitais, clínicas, centros de saúde e contactos úteis da Cidade do Mindelo. A aplicação SOS Mindelo, também é capaz de realizar chamadas e envio de emails para entidades de emergências. As interfaces de utilizador foram projectadas com o uso da linguagem XML e o código fonte foi escrito com uso da linguagem de programação Java.

2.3.2 Estrutura do projecto SOS Mindelo

As aplicações Android no geral obedecem uma estrutura similar. A estrutura da aplicação SOS Mindelo apresenta-se na figura 32.

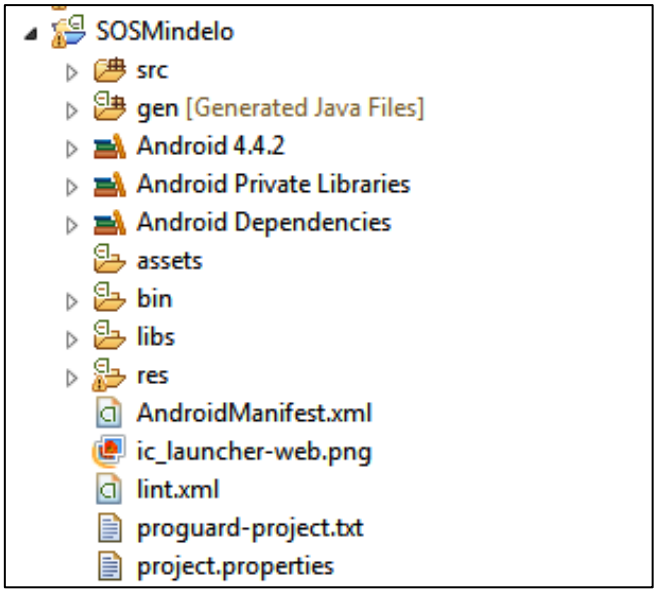


Figura 32: Estrutura da aplicação SOS Mindelo

A aplicação SOS Mindelo possui os seguintes componentes:

- **src**: A pasta src armazena a Activity principal e todas as outras classes que compõe o código fonte Java de uma aplicação em Android. A figura 33 mostra a Activity principal denominada de MainActivity e os demais ficheiros que compõe o código fonte Java da aplicação SOS Mindelo;

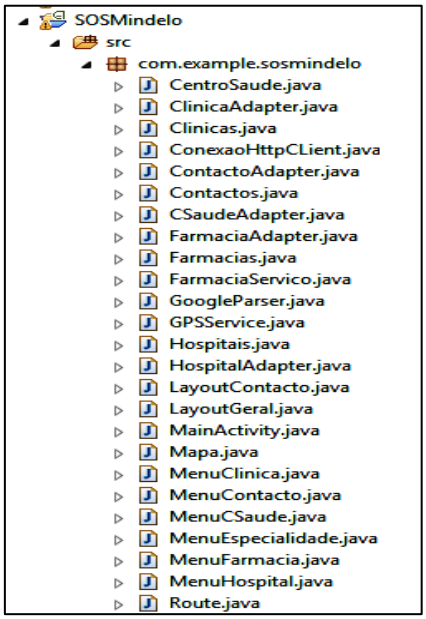


Figura 33: Pasta Source da aplicação SOS Mindelo

- **gen**: A pasta gen armazena os arquivos fontes que são gerados automaticamente pela plataforma Android. Estes arquivos não podem ser alterados pelo desenvolvedor. Esta pasta dispõe da classe **R.java** que é criada automaticamente e contém todas as constantes que representam os recursos de aplicação. Por exemplo, quando se declara um botão em um arquivo XML, automaticamente a classe R.java cria a referência desse botão, que poderá ser acedido através dessa referência dentro do código fonte;
- **assets**: A pasta assets armazena os arquivos mídia usados na aplicação. Os arquivos da pasta assets são acessíveis somente durante a programação. Como exemplo, quando se pretende personalizar fontes na aplicação, o ficheiro responsável por personalizar a fonte deve ser armazenada nesta pasta;
- **bin**: A pasta bin armazena os arquivos gerados automaticamente pela plataforma de desenvolvimento;
- **libs**: A pasta libs encontra as bibliotecas adicionais utilizadas pela aplicação;

- **res**: A pasta res contém os recursos utilizados na aplicação como as interfaces, imagens, estilos. Esta pasta encontra subdividida nas seguintes pastas, com propósitos específicos:
  - **drawable**: As pastas com o sufixo drawable são guardadas todas as imagens usadas no projecto;
  - **values**: A pasta values também é destinada para os arquivos XML que contem diferentes atributos como dimensões, estilos, strings, etc.
  - **layout**: A pasta layout onde estão os arquivos no formato XML que representam as interfaces de utilizador da aplicação. A figura 34 mostra a pasta layout;

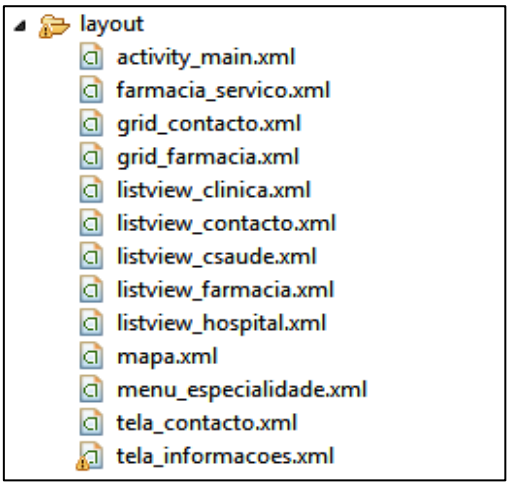


Figura 34: Pasta resource da aplicação SOS Mindelo

- **AndroidManifest.xml**: O AndroidManifest.xml é um arquivo de extrema importância para o projecto. Neste arquivo é feita toda a configuração dos recursos utilizados na aplicação. A figura 35 mostra o arquivo AndroidManifest.xml;

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sosmindelo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="19" />

     <permission
        android:name="com.example.sosmindelo.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />
     <uses-permission android:name="com.example.sosmindelo.permission.MAPS_RECEIVE"/>
     <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
     <uses-permission android:name="android.permission.INTERNET"/>
     <uses-permission android:name="android.permission.CALL_PHONE"/>
```

Figura 35: Arquivo AndroidMnifest.xml

2.3.3 A Base de Dados da aplicação

Para o armazenamento dos dados referentes às farmácias, hospitais, clínicas, centos de saúde e contactos úteis foi utilizado o SGBD MySql para a criação de uma base de dados. A base de dados encontra-se armazenada num servidor *online* do provedor de hospedagem Hostinger. As tabelas da base de dados podem ser vistos no diagrama apresentado na figura 36, e não há relacionamentos entre as tabelas.

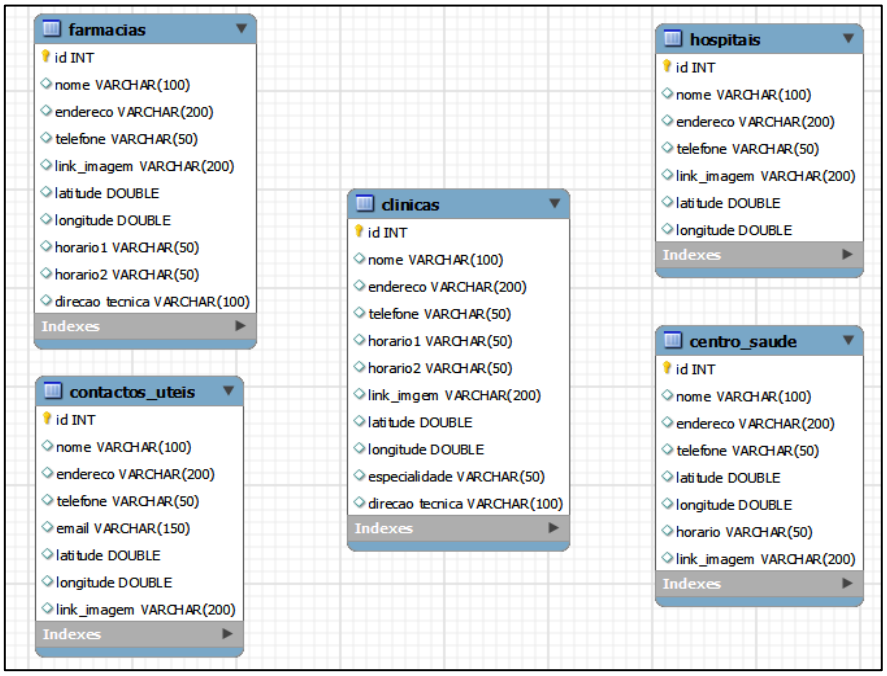


Figura 36: Tabelas da base de dados

2.3.4 Conexão á base de dados Online

Para a busca de informações na base de dados se utilizam o HTTP e o recurso JSON [42] para a importação dos dados para o Android.

Para a conexão entre a base de dados e a aplicação Android, primeiramente a aplicação faz a chamada do Script PHP correspondente aos dados requisitados, com o uso dos métodos Post e Get do protocolo HTTP. Os dados recebidos são então codificados em formato JSON pelo mesmo script possibilitando assim a sua interpretação por parte da aplicação Android. Na figura 37 pode ver-se o processo de conexão entre uma aplicação Android e uma base de dados MySql.

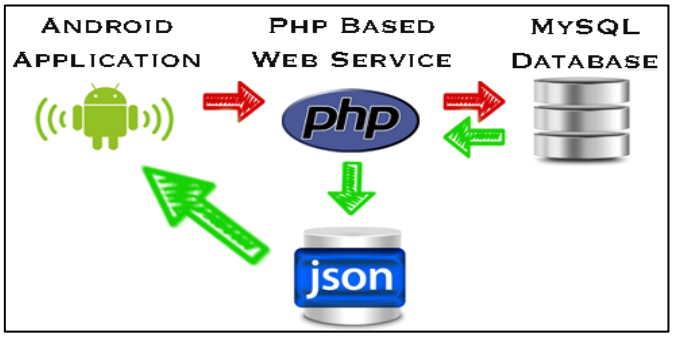


Figura 37: Conexão entre o Android e Base dados MySql [43]

As figuras 38 e 39 apresentam o extracto de código que mostra como a conexão entre a base de dados e a aplicação Android é feita e como os dados no formato JSON são convertidos para String.

```
String resposta;
try{
    // Armazena o resultado retornado pelo script PHP que resulta da consulta MySQL
    resposta=ConexaoHttpClient.executeHttpGet("http://sosmindelo.esy.es/farmacias.php");
    String result=resposta.toString();
```

Figura 38: Conexão com a Base de dados externa

```
try{

    JSONArray jArray = new JSONArray(result);
    for(int i=0;i<jArray.length();i++){
        JSONObject json_data = jArray.getJSONObject(i);
```

Figura 39: Dados no formato JSON convertidos para String

2.3.5 Serviços de Localização

No Android também é possível desenvolver aplicações baseadas em localizações. Os serviços de localização permitem determinar a localização do dispositivo na superfície da terra e estão disponíveis através do uso de recursos encontrados nas classes do pacote android.location.

No desenvolvimento da aplicação foi criada uma classe denominada de GPSService que usa as classes do pacote android.location para determinar a localização do dispositivo. Para a implementação da classe GPSService foram utilizados as seguintes classes do pacote android.location:

- **LocationManager**;
- **LocationProvider**;
- **Location**.

A **classe LocationManager** (44) permite acessar os serviços de localização no dispositivo. Através desta classe se pode obter a posição actual do dispositivo e obter actualizações quando o dispositivo estiver em movimento.

A classe LocationManager não pode ser instanciada directamente. É necessário utilizar o método seguinte para se obter uma instância dessa classe:

- *Context.getSystemService(Context.LOCATION SERVICE).*

A **classe LocationProvider** (45) permite especificar o provedor de localização que fornece as informações acerca da posição actual. Estão disponíveis os seguintes LocationProviders:

- *GPS_PROVIDER*: Determina a localização via satélites através do GPS do dispositivo. Apresenta uma maior precisão, no entanto o consumo de bateria é alto;
- *NETWORK_PROVIDER*: Utiliza a rede móvel ou Wi-Fi para determinar a melhor localização. A sua precisão é menor que a GPS, mas é mais rápida e apresenta menor consumo da bateria;
- *PASSIVE_PROVIDER*: Determina a localização de forma passiva, quando outros aplicativos ou serviços fazem a solicitação.

A **classe Location** [46] permite solicitar a localização do dispositivo. A localização do dispositivo pode conter informações de latitude, longitude, altitude, entre outros.

Para ter acesso aos serviços de localização na aplicação primeiramente, é necessário ter permissão no ficheiro AndroidManifest.xml, como mostra a figura 40

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Figura 40: Permissão para uso de GPS

Na figura 41 mostra um extracto de código fonte do método **getLocation** da classe GPSService onde se pode ver a implementação das classes acima referida.

```
public Location getLocation(){
    try{
        lm=(LocationManager)mContext.getSystemService(LOCATION_SERVICE);
        // obtendo  o  estado  do GPS
        isGPSEnable=lm.isProviderEnabled(LocationManager.GPS_PROVIDER);
        if(!isGPSEnable){

        }else{
            this.canGetLocation=true;
             // Se GPS estiver ativado obtem as coordenadas de latitude e longitude usando os Servicos d
             if (isGPSEnable) {
                    if (location == null) {
                        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,1000 * 60 * 1,10, this);
                        Log.d("GPS Enabled", "GPS Enabled");
                        if (lm != null) {
                            location = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                            if (location != null) {
                                latitude = location.getLatitude();
                                longitude = location.getLongitude();
```

Figura 41: Método getLocation da classe GPSService

2.3.6 API Google Maps

O API Google Maps [47] é um serviço gratuito, desenvolvido pelo Google, que permite a pesquisa e a visualização de mapas de ruas, imagens de satélite da terra, bem como utilizar as funções para o desenho de rotas.

Para utilizar o API Google Maps é necessário estar registado com o serviço do Google Maps, concordar com os termos de serviço e obter uma chave de autenticação. O API Google Maps é utilizado para possibilitar aos utilizadores a visualização de locais e rotas de destino. Para ter acesso ao API do Google Maps e necessário ter acesso a determinadas permissões, que se apresentam na figura 42.

```
<permission
   android:name="com.example.sosmindelo.permission.MAPS_RECEIVE"
   android:protectionLevel="signature" />
<uses-permission android:name="com.example.sosmindelo.permission.MAPS_RECEIVE"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

Figura 42: Permissão no AndroidManifest.xml para obter o Google Maps

Para visualizar o mapa na tela e determinar a localização do utilizador e dos lugares, foi criada a classe Mapa que implementa a classe SupportMapFragment [48]. A classe SupportMapFragment é responsável por colocar um mapa em uma aplicação. As figuras 43 e 44 mostram os códigos fontes em linguagem Java e em linguagem xml usados para implementar a classe SupportMapFragment.

```
SupportMapFragment fragment=(SupportMapFragment)getSupportFragmentManager().
        findFragmentById(R.id.maps);
mapa=fragment.getMap();
```

Figura 43: Código em Java da classe SupportMapFragment

```
<fragment
    android:id="@+id/maps"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true" />
```

Figura 44: Código xml da classe SupportMapFragment

O utilizador e os lugares presentes na aplicação são identificados no mapa através do uso do recurso Marker [49]. O Marker representa um ícone que é colocado num ponto particular da superfície do mapa. A figura 45 mostra um extracto de código que é responsável pela marcação no mapa da posição do utilizador e do lugar pretendido pelo utilizador.

```
LatLng cord_utilizador = new LatLng(latGPS,longGPS);
LatLng cord_local = new LatLng(lat , lg);
if(mapa!=null){
      Marker utilizador = mapa.addMarker(new MarkerOptions().position(cord_utilizador).
            icon(BitmapDescriptorFactory.fromResource(R.drawable.eu)).title("Eu"));
      Marker lugar = mapa.addMarker(new MarkerOptions().position(cord_local).icon(
            BitmapDescriptorFactory.fromResource(R.drawable.lugar)).title(nome).snippet(endereco));
      utilizador.showInfoWindow();
      lugar.showInfoWindow();
  }
```

Figura 45: Marcação dos pontos no mapa

Para a definição da rota foi criada dentro da classe Mapa, a classe ProcessoAsync que estende a Async Task. Também foi criada as seguintes classes:

- Route.java;
- GoogleParser.java;

A classe **ProcessoAsync** é responsável por acessar ao *site* do Google Maps e por baixar o JSON com a informação da rota. A figura 46 mostra um extracto do código fonte responsável para baixar o JSON com a informação da rota.

```
private Route directions(final LatLng start, final LatLng dest) {

    // Formatando a URL com a latitude e longitude
    // de origem e destino.
    String urlRota = String.format(Locale.US,
            "http://maps.googleapis.com/maps/api/"+
            "directions/json?origin=%f,%f&"+
            "destination=%f,%f&" +
            "sensor=false&mode=driving",
            start.latitude,
            start.longitude,
            dest.latitude,
            dest.longitude);

    GoogleParser parser;
    parser = new GoogleParser(urlRota);
    return parser.parse();
}
```

Figura 46: Classe ProcessoAsync

A classe **Route.java** cria uma lista de coordenadas geográficas latitude e longitude. A figura 47 mostra um extracto da implementação da classe Route.java;

```
public class Route {
      private final List<LatLng> points;
      private String polyline;

      public Route() {
        points = new ArrayList<LatLng>();
      }

      public void addPoints(final List<LatLng> points) {
        this.points.addAll(points);
      }
```

Figura 47: Classe Route.java

A classe **GoogleParser.java** através da URL com a informação da rota pega a primeira rota a ser tomada e os passos do caminho. Após isso a classe GoogleParser.java decodifica o polyline e adiciona a rota. A figura 48 mostra um extracto de código da classe GoogleParser.java;

```
// Pega a primeira rota retornada
JSONObject jsonRoute =json.getJSONArray("routes").getJSONObject(0);
JSONObject leg = jsonRoute.getJSONArray("legs").getJSONObject(0);
// Obtém os passos do caminho
JSONArray steps = leg.getJSONArray("steps");

final int numSteps = steps.length();
/*
 * Itera através dos passos, decodificando
 * a polyline e adicionando à rota.
 */
JSONObject step;
for (int i = 0; i < numSteps; i++) {
  // Obtém o passo corrente
  step = steps.getJSONObject(i);
  // Decodifica a polyline  atraves do metodo decodePolyline e adiciona à rota
  route.addPoints(decodePolyLine(step.getJSONObject("polyline").getString("points")));
```

Figura 48: Classe GoogleParser.java

2.4 Descrição das Activities

Neste tópico serão apresentadas as activities da aplicação e as suas respectivas funcionalidades para melhor ilustrar o funcionamento da aplicação geral. Cada interface de utilizador representa uma Activity. A interface do utilizador foi criada de modo a facilitar a compreensão e navegação na aplicação por parte dos utilizadores. Outras interfaces de utilizador da aplicação podem ser consultadas no anexo II.

Na figura 49 mostra-se o logótipo da aplicação SOS Mindelo.

Figura 49: Logótipo da aplicação SOS Mindelo

2.4.1 Activity MainActivity

A Activity MainActivity é a Activity principal da aplicação. Esta Activity contém uma interface com um menu inicial da aplicação composta pelos botões Farmácias, Hospitais, Clinicas, Centros Saúde, Contactos Úteis. A Activity principal é utilizada para carregar a interface inicial da aplicação.

A figura 50 mostra a interface de utilizador da Activity principal da aplicação e o respectivo código fonte em linguagem xml.

Figura 50: Interface da Activity Principal

A navegação para as outras interfaces de utilizador é feita através da interface principal. Quando um botão é accionado, a Activity apropriada é inicializada através do componente Intent.

As Figuras 51, 52, 53, 54 e 55 mostram como o evento de cada botão é implementado.

```
// Acao do butao farmacia
btn_farmacia=(Button)findViewById(R.id.btn_farmacia);
btn_farmacia.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        Intent farmacia = new Intent(MainActivity.this,MenuFarmacia.class);
        farmacia.putExtras(coordenadas);
        startActivity(farmacia);

    }
});
```

Figura 51: Evento do botão Farmácias

```
// Acao do butao hospital
btn_hospital=(Button)findViewById(R.id.btn_hospital);
btn_hospital.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        Intent hospital=new Intent(MainActivity.this,MenuHospital.class);
        hospital.putExtras(coordenadas);
        startActivity(hospital);

    }
});
```

Figura 52: Evento do botão Hospitais

```
// Acao do butao clinica
btn_clinica=(Button)findViewById(R.id.btn_clinica);
btn_clinica.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        Intent clinica=new Intent(MainActivity.this,MenuEspecialidade.class);
        clinica.putExtras(coordenadas);
        startActivity(clinica);
    }
});
```

Figura 53: Evento do botão Clinicas

```
// Acao do butao centro saude
btn_csaude=(Button)findViewById(R.id.btn_csaude);
btn_csaude.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        Intent csaude=new Intent(MainActivity.this,MenuCSaude.class);
        csaude.putExtras(coordenadas);
        startActivity(csaude);
    }
});
```

Figura 54: Evento do botão Centros Saúde

```
// Acao do butao contactos
btn_contacto=(Button)findViewById(R.id.btn_contacto);
btn_contacto.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        Intent contacto=new Intent(MainActivity.this,MenuContacto.class);
        contacto.putExtras(coordenadas);
        startActivity(contacto);

    }
});
```

Figura 55: Evento do botão Contactos Úteis

2.4.2 Activity MenuFarmacia

A Activity MenuFarmacia é inicializada quando o botão Farmácias da Activity principal for seleccionado. Esta Activity implementa uma ListView onde se podem ver as farmácias da cidade do Mindelo. Cada item da ListView é composto pelo nome da farmácia, ícone, localização e a distância em metros entre o utilizador e a farmácia em questão no momento solicitado.

As farmácias são exibidas na lista por ordem crescente de distância. A primeira farmácia a ser exibida apresenta o menor valor de distância, ou seja, representa a farmácia mais próxima do utilizador, a segunda farmácia representa a segunda farmácia mais próxima e assim sucessivamente.

A figura 56 mostra a interface de utilizador da Activity MenuFarmacia.

Figura 56: Interface da Activity MenuFarmacia

2.4.3 Activity FarmaciaServico

A Activity FarmaciaServico inicializa quando é accionado o botão Farmácia de Serviço, na Activity MenuFarmacia. A Activity FarmaciaServico apresenta ao utilizador uma página Web com as farmácias e as respectivas datas em que estão de serviço. Estas informações são requisitadas no site "www.portondinosilha.cv", com uso da sua URL.

Para carregar a URL foi utilizado o WebView [50]. O WebView é um elemento da classe android.webkit que permite exibir páginas Web dentro de uma aplicação Android. A figura 57 exibe a interface de utilizador da Activity FarmaciaServico.

Figura 57: Interface da Activity FarmaciaServico

2.4.4 Activity MenuEspecialidade

A Activity MenuEspecialidade inicia quando o botão Clínicas é seleccionado na Activity MainActivity. Esta Activity apresenta ao utilizador um submenu composto pelos botões Geral, Odontologia, Oftalmologia, Ginecologia, Endoscopia, Fisioterapia e Analises Clínicas. Estes botões representam as especialidades existentes de clinicas na cidade do Mindelo. A figura 58 mostra a interface de utilizador desta Activity.

Figura 58: Interface da Activity MenuEspecialidade

A navegação para a Activity MenuClinica depende da opção escolhida pelo utilizador na Activity MenuEspecialidade. Quando um botão é accionado a Activity MenuClinica é inicializada através do componente Intent. As figuras 59 e 60 mostram os eventos dos botões Geral e Odontologia.

```
btn_geral=(Button)findViewById(R.id.btn_geral);
  btn_geral.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
          // TODO Auto-generated method stub
          Intent clinica= new Intent(MenuEspecialidade.this, MenuClinica.class);
          esp="Geral";
          cli_params.putString("especialidade", esp);
          clinica.putExtras(cli_params);
          startActivity(clinica);

      }
  });
```

Figura 59: Evento do botão Geral da Activity MenuEspecialidade

```
btn_odont=(Button)findViewById(R.id.btn_dentaria);
btn_odont.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        Intent clinica= new Intent(MenuEspecialidade.this, MenuClinica.class);
        esp="Odontologia";
        cli_params.putString("especialidade", esp);
        clinica.putExtras(cli_params);
        startActivity(clinica);

    }
});
```

Figura 60: Evento do botão Odontologia da Activity MenuEspecialidade

2.4.5 Activity MenuClinica

A Activity MenuClinica é inicializada quando um dos botões existentes na Activity MenuEspecialidade for accionado. Esta Activity implementa uma ListView onde se podem ver as Clinicas da cidade do Mindelo mediante o tipo escolhido pelo utilizador. Cada item da ListView é composto pelo nome da clinica, ícone, localização e a distância em metros entre o utilizador e a clinica em questão no momento solicitado.

As clinica são exibidas na lista por ordem crescente de distância. A primeira clinica a ser exibida apresenta o menor valor de distância e representa a clinica mais próxima do utilizador, a segunda clinica representa a segunda clinica mais próxima e assim sucessivamente.

A figura 61 mostra a interface de utilizador da Activity MenuClinica para o caso onde foi escolhida a opção Odontologia.

Figura 61: Interface da Activity MenuClinica

2.4.6 Activity MenuHospital

A Activity **MenuHospital** é inicializada quando o botão Hospitais na Activity MainActivity é seleccionado. Esta Activity implementa uma ListView onde se pode ver o Hospital da cidade do Mindelo. Cada item da ListView é composto pelo nome do Hospital, ícone, localização e a distância em metros entre o utilizador e o hospital no momento solicitado.

A figura 62 mostra a interface de utilizador da Activity MenuHospital.

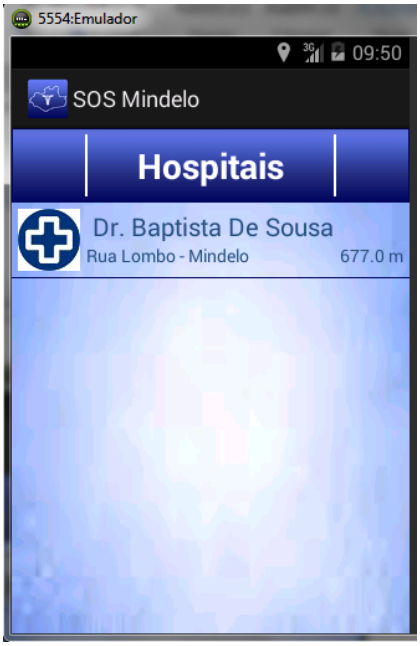


Figura 62: Interface da Activity MenuHospital

2.4.7 Activity MenuCSaude

A Activity **MenuCSaude** é inicializada quando o botão Centros Saúde na Activity MainActivity é seleccionado. Esta Activity implementa uma ListView onde se podem ver os centros de saúde da cidade do Mindelo. Cada item da ListView é composto pelo nome do centro de saúde, ícone, localização e a distância em metros entre o utilizador e o hospital em questão no momento solicitado.

Os centros de saúde são exibidos na lista por ordem crescente de distância. A semelhança do que ocorre na Activity na MenuFarmacia, o primeiro centro de saúde a ser exibido apresenta o menor valor de distância e representa o centro de saúde mais próximo

do utilizador, o segundo centro de saúde representa o segundo centro de saúde mais próximo e assim sucessivamente.

A figura 63 mostra a interface de utilizador da Activity MenuCSaude.

Figura 63: Interface da Activity MenuCSaude

2.4.8 Activity MenuContacto

A Activity **MenuContacto** é inicializada quando o botão Contactos Úteis da Activity MainActivity é seleccionado. Esta Activity implementa uma ListView onde se podem ver os contactos úteis da cidade do Mindelo. Cada item da ListView é composto pelo nome do contacto, imagem, localização e o número de telefone.

A figura 64 mostra a interface de utilizador da Activity MenuContacto.

Figura 64: Interface da Activity MenuContacto

2.4.9 Activity LayoutGeral

A Activity LayoutGeral inicializa num dos casos a seguir:

- Quando um item da ListView, apresentada pela Activity MenuFarmacia, for seleccionado;
- Quando um item da ListView, apresentada pela Activity MenuClinica, for seleccionado;
- Quando um item da ListView, apresentada pela Activity MenuHospital, for seleccionado;
- Quando um item da ListView, apresentada pela Activity MenuCSaude, for seleccionado;

A figura 65 mostra a interface de usuário da Activity LayoutGeral, neste caso foi seleccionado um item na lista de farmácia.

Figura 65: Interface da Activity LayoutGeral

A Activity LayoutGeral recebe as informações que são passadas como parâmetros, através do método getIntent e são posteriormente exibidas ao utilizador.

A Activity LayoutGeral é composta pelos botões a seguir:

- **Mapa:** O botão Mapa permite a visualização do mapa ao ser accionado;
- **Ligar**: O botão Ligar ao ser accionado implementa o evento responsável para realizar uma chamada ao lugar pretendido. A figura 66 mostra o código fonte do botão Ligar;

```
btn_ligar=(Button)findViewById(R.id.btn_chamada);
btn_ligar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        try {
            Intent intentCall = new Intent(Intent.ACTION_CALL);
            intentCall.setData(Uri.parse("tel:"+num_telefone));
            startActivity(intentCall);
            } catch (Exception e) {
            Log.e("Erro", "Falha na realizacao da chamada", e);
            }

    }
});
```

Figura 66: Evento do botão Ligar

2.4.10 Activity Mapa

A Activity Mapa inicia quando o botão Mapa da Activity LayoutGeral for seleccionado. A Activity Mapa permite ao utilizador saber o seu posicionamento no mapa quando consultar um dos lugares presentes na aplicação e obter uma rota de destino.

A figura 67 mostra a interface de utilizador da Activity Mapa.

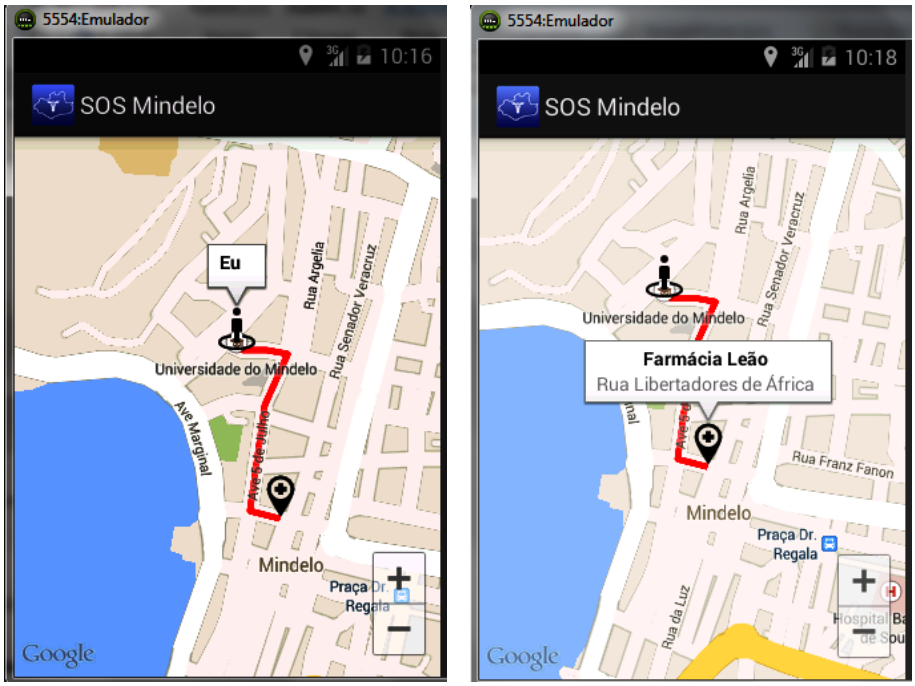


Figura 67: Interface da Activity Mapa

2.4.11 Activity LayoutContacto

A Activity LayoutContacto inicializa quando um item da ListView, apresentada pela Activity MenuContacto for seleccionado. Esta Activity também recebe as informações do contacto útil, que são passadas como parâmetros, através do método getIntent e são posteriormente exibidas ao utilizador.

A figura 68 mostra a interface de utilizador da Activity LayoutContacto.

Figura 68: Interface da Activity LayoutContacto

A Activity LayoutContacto é composta pelos botões:

- **Botão Ligar**: o botão Ligar ao ser accionado, realiza uma chamada para o número de emergência em questão. Este botão possui a mesma implementação do botão Ligar, mencionado na secção 2.5.9;
- **Botão Email**: o botão Email ao ser accionado, realiza o envio de email para a entidade de emergência em questão. O email é preenchido automaticamente com um texto com coordenadas geográficas do aparelho e um link para visualizar a posição do utilizador no mapa. A figura 69 mostra um extracto do código que implementa o evento do botão Email.

```
btn_email=(Button)findViewById(R.id.btn_email);
   btn_email.setOnClickListener(new View.OnClickListener() {

       @Override
       public void onClick(View v) {
           // TODO Auto-generated method stub
           try {   String[] TO = {emailAdress};
                   Intent intent = new Intent(Intent.ACTION_SEND);
                   intent.setType("plain/text");
                   intent.putExtra(Intent.EXTRA_SUBJECT, "SOS");// titulo  do  emial
                   intent.putExtra(Intent.EXTRA_TEXT, "Preciso  de ajuda as minhas coordenadas sao:"
                           + "\nLatitude: "+latitude+"\nLongitude: "+longitude+"\n\n"
                           +"clique no link abaixo determinar a minha localizacao no Mapa:\n\n https:"
                           + "//www.google.cv/maps/place/16%C2%B053'21.5%22N+24%C2%B059'"
                           + "20.0%22W/@"+latitude+","+longitude+",17z/data=!4m2!3m1!1s0x0:0x0?hl=pt-PT");
                   intent.putExtra(Intent.EXTRA_EMAIL, TO);// destinatario
                   startActivity(Intent.createChooser(intent, "Enviar Email"));
               } catch (Exception e) {
               Log.e("Erro", "Falha no  envio  de email", e);
               }

       }
   });
```

Figura 69: Evento do botão Email

2.5 Scripts PHP

Os Scripts PHP são responsáveis por estabelecer a conexão com a Base de dados. A figura 70 apresenta um extracto de código do script PHP que obtém da base de dados os dados de todas as farmácias. Os códigos dos outros Scripts podem ser consultados no Anexo I.

```php
<?php

 // Variavel PHP para armaznar o  endereco do Host
$host = "mysql.hostinger.com.br";
// Variavel PHP para armazenar o userID
$userID  = "u293868726_leo";
 // Variavel PHP para armazenar a palavra-passe
$password = "********";
//  // Variavel PHP para armazenar o  nome da base de dados
$BD  = "u293868726_sos";
 // Variavel PHP pata armazenar o  resultado da funcao PHP  'mysql_connect()'
// que estabelece a concecao entre PHP & MySQL
$db_con = mysql_connect($host,$userID,$password) or die('Impossivel estabelecer a comunicacao');
mysql_select_db($BD);
$sql = "SELECT * FROM farmacias";
$result = mysql_query($sql);
$json = array();

while($row=mysql_fetch_assoc($result))
 $output[]=$row;
print(json_encode($output));
mysql_close();
?>
```

Figura 70: Script PHP responsável para obter dados das Farmácias na Base de dados

Conclusão

Neste trabalho foi projectada e desenvolvida uma aplicação para dispositivos móveis com o sistema Android, através do uso das tecnologias principalmente os serviços de localização (GPS), Google Maps API e das tecnologias *Open Source* (MySql e PHP) todos os objectivos inicialmente estipulados neste trabalho foram cumpridos, nomeadamente é possível:

- Localizar todas as farmácias, hospitais, clínicas e centros de saúde da cidade do Mindelo mais próximas do utilizador no momento solicitado, utilizando a localização geográfica do dispositivo;
- Informar o utilizador sobre a localização, horário de funcionamento, contacto e o director farmacêutico das farmácias;
- Informar sobre as farmácias de serviço da cidade do Mindelo;
- Informar o utilizador sobre a localização e o contacto do hospital;
- Informar o utilizador sobre a localização, horário de funcionamento, director técnico e o contacto das clínicas;
- Informar o utilizador sobre as especialidades de cada clinica;
- Informar o utilizador sobre a localização, horário de funcionamento e o contacto dos centros de saúde;
- Exibir as entidades de emergências da cidade do Mindelo.
- Determinar no mapa a localização dos lugares presentes na aplicação e do utilizador;
- Exibir no mapa, uma rota de destino entre o utilizador e o lugar escolhido tendo em conta as suas posições geográficas;
- Efectuar ligações telefónicas aos locais presentes na aplicação com uso da própria aplicação;
- Envio de email para as entidades de emergências com uso da própria aplicação;
- Efectuar chamadas para números de emergências;

Após ter feito os testes em um *smartphone* foi possível comprovar todas as funcionalidades da aplicação. Através da utilização da aplicação SOS Mindelo, tem-se uma ferramenta capaz de disponibilizar informações em caso de emergências de saúde na cidade do Mindelo.

Trabalhos Futuros

Como trabalho futuro, se pretende:

- Estender a aplicação para dispositivos com tela maiores (Tablets);
- Desenvolver a aplicação para dispositivos com o sistema IOS;
- Estender a aplicação para as outras ilhas;
- Optimizar o mapa de modo a disponibilizar mais que uma rota.

Bibliografia

1. **Vital, Guilherme.** Tecnoinfo. *http://atecnoinfo.blogspot.com/.* [Online] 2012 de Novembro de 16. [Citação: 8 de Março de 2014.] http://atecnoinfo.blogspot.com/2012/11/a-evolucao-dos-celulares.html.

2. **Samira Furtado.** Grupo CVTelecom. *http://www.grupocvt.com.cv/.* [Online] 14 de Novembro de 2011. [Citação: 8 de Março de 2013.] http://www.grupocvt.com.cv/node/306.

3. **Cardoso, Eduardo.** legalafrica. *http://legalafrica.wordpress.com.* [Online] 26 de Julho de 2012. [Citação: 8 de Março de 2014.] http://legalafrica.wordpress.com/2012/07/26/cabo-verde-introducao-do-3g-triplica-o-numero-de-assinantes-de-internet/.

4. **Farmacias de Servico.net.** www.farmaciasdeservico.net. [Online] [Citação: 19 de Outubro de 2014.] www.farmaciasdeservico.net/e/web/andoid.

5. **Peer Farmacie sas.** Play Store. *https://play.google.com/store.* [Online] 7 de Outubro de 2014. [Citação: 19 de Outubro de 2017.] https://play.google.com/store/apps/details?id=wla.com.apo.

6. Play Store. *https://play.google.com.* [Online] [Citação: 19 de Outubro de 2014.] https://play.google.com/store/apps/details?id=ma.casanet.PharmaGarde.

7. Play Store. *https://play.google.com/.* [Online] 13 de Janeiro de 2014. [Citação: 7 de Outubro de 2014.] https://play.google.com/store/apps/details?id=net.cofb.android.farmaguia.

8. Play Store. *https://play.google.com/.* [Online] [Citação: 30 de Janeiro de 2015.] https://play.google.com/store/apps/details?id=com.apps.marpharma.

9. wikipedia. *http://pt.wikipedia.org/.* [Online] [Citação: 20 de Outubro de 2014.] http://pt.wikipedia.org/wiki/Android.

10. wikipedia. *http://pt.wikipedia.org/.* [Online] [Citação: 20 de Outubro de 2014.] http://pt.wikipedia.org/wiki/Open_Handset_Alliance.

11. IDC. *http://www.idc.com.* [Online] 2014. [Citação: 2015 de Fevereiro de 8.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

12. **Aislan.** scribd. *http://www.scribd.com.* [Online] 1 de Junho de 2009. [Citação: 20 de Outubro de 2014.] http://www.scribd.com/doc/15981338/Introducao-a-Plataforma-Android.
13. Wikipedia. *http://pt.wikipedia.org.* [Online] [Citação: 9 de Fevereiro de 2015.] http://pt.wikipedia.org/wiki/Hist%C3%B3rico_de_vers%C3%B5es_do_Android.
14. **Renato Santino.** Olhar Digital. *http://olhardigital.uol.com.br/home.* [Online] 11 de Julho de 2013. [Citação: 9 de Fevereiro de 2015.] http://olhardigital.uol.com.br/noticia/android-ja-teve-10-versoes-diferentes-relembre-a-evolucao-do-sistema/35801.
15. android. *http://www.android.com.* [Online] [Citação: 18 de Fevereiro de 2015.] http://www.android.com/intl/pt-BR_br/versions/kit-kat-4-4/.
16. gordelicias. *http://gordelicias.biz.* [Online] [Citação: 18 de Fevereiro de 2015.] http://gordelicias.biz/index.php/2013/09/04/android-kitkat-uma-parceria-google-e-nestle/.
17. android. *http://www.android.com.* [Online] [Citação: 18 de Fevereiro de 2015.] http://www.android.com/intl/pt-BR_br/versions/lollipop-5-0/.
18. Android Developers Blog. *http://android-developers.blogspot.com.* [Online] [Citação: 18 de Fevereiro de 2015.] http://android-developers.blogspot.com/2014/10/android-50-lollipop-sdk-and-nexus.html.
19. Developer Android. *https://developer.android.com.* [Online] [Citação: 20 de Fevereiro de 2015.] https://developer.android.com/about/dashboards/index.html.
20. **Strickland, Jonathan.** Como tudo Funciona. *http://www.hsw.uol.com.br/.* [Online] [Citação: 22 de Outubro de 2014.] http://tecnologia.hsw.uol.com.br/google-phone2.htm.
21. developer.android. *http://developer.android.com.* [Online] [Citação: 22 de Outubro de 2014.] http://developer.android.com/images/system-architecture.jpg.
22. Developer Android. *http://developer.android.com/.* [Online] [Citação: 27 de Outubro de 2014.] http://developer.android.com/guide/components/fundamentals.html.
23. **Tosin, Carlos.** Dicas-l. *http://www.dicas-l.com.br/.* [Online] 3 de Maio de http://www.dicas-l.com.br/. [Citação: 27 de Outubro de 2014.] http://www.dicas-l.com.br/arquivo/conhecendo_o_android.php#.VK55yyswq6y.
24. Developer Android. *http://developer.android.com/.* [Online] [Citação: 27 de Outubro de 2014.] http://developer.android.com/guide/components/activities.html.

25. Developer Android. *http://developer.android.com/.* [Online] [Citação: 29 de Outubro de 2014.] http://developer.android.com/guide/components/services.html.
26. Developer Android. *http://developer.android.com/.* [Online] [Citação: 29 de Outubro de 2014.] http://developer.android.com/guide/topics/providers/content-providers.html.
27. Developer Android. *http://developer.android.com/.* [Online] [Citação: 29 de Outubro de 2014.] http://developer.android.com/guide/topics/manifest/manifest-intro.html.
28. Developer Android. *http://developer.android.com/.* [Online] [Citação: 30 de Outubro de 2014.] http://developer.android.com/guide/components/intents-filters.html.
29. Developer Android. *http://developer.android.com/.* [Online] [Citação: 30 de Outubro de 2014.] http://developer.android.com/reference/android/view/View.html.
30. Developer Android. *http://developer.android.com.* [Online] [Citação: 10 de Janeiro de 2015.] http://developer.android.com/reference/android/widget/package-summary.html.
31. Developer Android. *http://developer.android.com/.* [Online] [Citação: 4 de Novembro de 2014.] http://developer.android.com/reference/android/app/Activity.html.
32. **Vogel, Lars.** Developer. *http://developer.android.com/.* [Online] [Citação: 11 de Novembro de 2014.] http://developer.android.com/tools/sdk/tools-notes.html.
33. wikipedia. *http://pt.wikipedia.org.* [Online] [Citação: 11 de Novembro de 2014.] http://pt.wikipedia.org/wiki/Eclipse_%28software%29.
34. Developer Android. *http://developer.android.com.* [Online] [Citação: 11 de Novembro de 2014.] http://developer.android.com/tools/help/adt.html#tools.
35. JavaMan. *http://www.javaman.com.br.* [Online] [Citação: 11 de Novembro de 2014.] http://www.javaman.com.br/artigos/versoesJava.html.
36. Docs Kde. *https://docs.kde.or.* [Online] [Citação: 12 de Fevereiro de 2015.] https://docs.kde.org/stable/pt_BR/kdesdk/umbrello/uml-basics.html.
37. Developer Android. *http://developer.android.com.* [Online] [Citação: 24 de Fevereiro de 2015.] http://developer.android.com/reference/android/net/ConnectivityManager.html.
38. Developer Android. *http://developer.android.com.* [Online] [Citação: 24 de Fevereiro de 2015.] http://developer.android.com/reference/android/os/AsyncTask.html.

39. Wikipedia. *http://pt.wikipedia.org.* [Online] [Citação: 23 de Janeiro de 2015.] http://pt.wikipedia.org/wiki/Teorema_de_Pit%C3%A1goras.
40. MapaRadar. *http://forum.maparadar.com.* [Online] 18 de Marco de 2011. [Citação: 23 de Janeiro de 2015.] http://forum.maparadar.com/viewtopic.php?f=145&t=5181.
41. Developer Android. *http://developer.android.com.* [Online] [Citação: 24 de Fevereiro de 2015.] http://developer.android.com/reference/android/widget/BaseAdapter.html.
42. JSON Org. *http://www.json.org/.* [Online] [Citação: 8 de Dezembro de 2014.] http://www.json.org/json-pt.html.
43. Mybringback. *http://www.mybringback.com.* [Online] 27 de Maio de 2013. [Citação: 20 de Abril de 2015.] http://www.mybringback.com/android-sdk/12924/android-tutorial-using-remote-databases-php-and-mysql-part-1/.
44. Developer Android. *http://developer.android.* [Online] [Citação: 9 de Dezembro de 2014.] http://developer.android.com/reference/android/location/LocationManager.html.
45. Developer Android. *http://developer.android.com.* [Online] [Citação: 9 de Dezembro de 2014.] http://developer.android.com/reference/android/location/LocationProvider.html.
46. Developer Android. *http://developer.android.com.* [Online] [Citação: 9 de Dezembro de 2014.] http://developer.android.com/reference/android/location/Location.html.
47. Developers Google. *https://developers.google.com.* [Online] [Citação: 9 de Dezembro de 2014.] https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2.
48. Developer Android. *http://developer.android.com.* [Online] [Citação: 24 de Janeiro de 2015.] http://developer.android.com/reference/com/google/android/gms/maps/SupportMapFragment.html.

49. Developer Android. *https://developer.android.* [Online] [Citação: 12 de Janeiro de 2015.] https://developer.android.com/reference/com/google/android/gms/maps/model/Marker.html.

50. Developer Android. *http://developer.android.com.* [Online] [Citação: 24 de Janeiro de 2015.] http://developer.android.com/reference/android/webkit/WebView.html.

Anexos

Anexo 1

O anexo 1 mostra os scripts PHP responsáveis para realizar a conexão com a base de dados.

1.1 Script PHP que obtém os dados do hospital da cidade do Mindelo.

```php
<?php

 // Variavel PHP para armaznar o  endereco do Host
$host = "mysql.hostinger.com.br";
// Variavel PHP para armazenar o userID
$userID  = "u293868726_leo";
 // Variavel PHP para armazenar a palavra-passe
$password = "********";
//  // Variavel PHP para armazenar o  nome da base de dados
$BD  = "u293868726_sos";
   // Variavel PHP pata armazenar o  resultado da funcao PHP  'mysql_connect()'
   //que estabelece a concecao entre PHP & MySQL
$db_con = mysql_connect($host,$userID,$password) or die('Impossivel estabelecer a comunicacao');
mysql_select_db($BD);
$sql = "SELECT * FROM hospitais";
$result = mysql_query($sql);
$json = array();

while($row=mysql_fetch_assoc($result))
 $output[]=$row;
print(json_encode($output));
mysql_close();
?>
```

1.2 Script PHP que obtém os dados de todos os centros de saúde existentes cidade do Mindelo.

```php
<?php

 // Variavel PHP para armaznar o  endereco do Host
$host = "mysql.hostinger.com.br";
// Variavel PHP para armazenar o userID
$userID  = "u293868726_leo";
 // Variavel PHP para armazenar a palavra-passe
$password = "********";
//  // Variavel PHP para armazenar o  nome da base de dados
$BD  = "u293868726_sos";
   // Variavel PHP pata armazenar o  resultado da funcao PHP  'mysql_connect()'
   //que estabelece a concecao entre PHP & MySQL
$db_con = mysql_connect($host,$userID,$password) or die('Impossivel estabelecer a comunicacao');
mysql_select_db($BD);
$sql = "SELECT * FROM centro_saude";
$result = mysql_query($sql);
$json = array();

while($row=mysql_fetch_assoc($result))
 $output[]=$row;
print(json_encode($output));
mysql_close();
?>
```

1.3 Script PHP que obtém os dados das clinicas existentes na cidade do Mindelo, mediante especialidade.

```php
<?php

 // Variavel PHP para armaznar o  endereco do Host
$host = "mysql.hostinger.com.br";
// Variavel PHP para armazenar o userID
$userID  = "u293868726_leo";
 // Variavel PHP para armazenar a palavra-passe
$password = "**********";
//  // Variavel PHP para armazenar o  nome da base de dados
$BD  = "u293868726_sos";
    // Variavel PHP pata armazenar o  resultado da funcao PHP  'mysql_connect()'
   //que estabelece a concecao entre PHP & MySQL
$db_con = mysql_connect($host,$userID,$password) or die('Impossivel estabelecer a comunicacao');
mysql_select_db($BD);
$sql = "SELECT * FROM clinicas WHERE especialidade  = '". $_POST["birthyear"]."'";
$result = mysql_query($sql);
$json = array();

while($row=mysql_fetch_assoc($result))
 $output[]=$row;
print(json_encode($output));
mysql_close();
?>
```

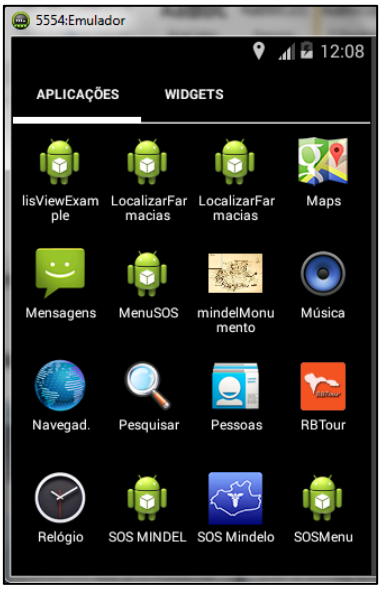
1.4 Script PHP que obtém os dados referentes aos contactos úteis.

```php
<?php

 // Variavel PHP para armaznar o  endereco do Host
$host = "mysql.hostinger.com.br";
// Variavel PHP para armazenar o userID
$userID  = "u293868726_leo";
 // Variavel PHP para armazenar a palavra-passe
$password = "********";
//  // Variavel PHP para armazenar o  nome da base de dados
$BD  =  "u293868726_sos";
    // Variavel PHP para armazenar o  resultado da funcao PHP  'mysql_connect()'
   //que estabelece a concecao entre PHP & MySQL
$db_con = mysql_connect($host,$userID,$password) or die('Impossivel estabelecer a comunicacao');
mysql_select_db($BD);
$sql = "SELECT * FROM contactos_uteis";
$result = mysql_query($sql);
$json = array();

while($row=mysql_fetch_assoc($result))
 $output[]=$row;
print(json_encode($output));
mysql_close();
?>
```

Anexo 2

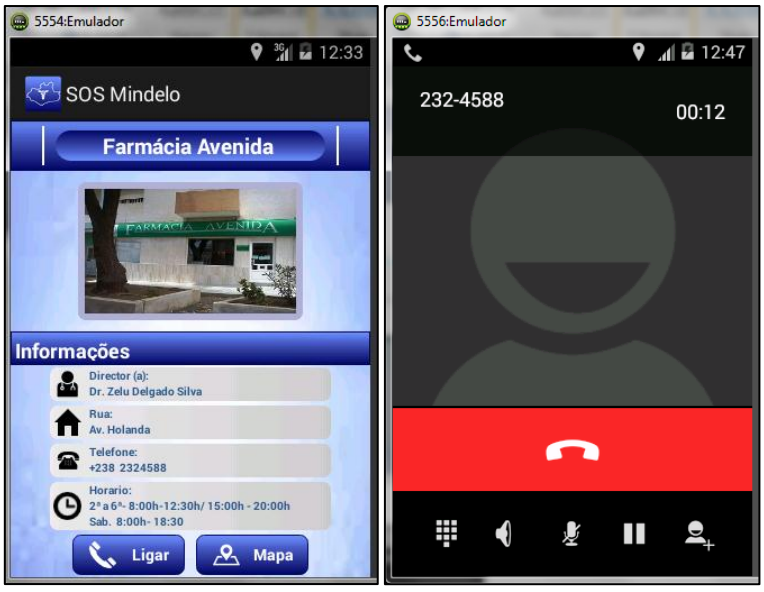
O anexo 2 mostra algumas interfaces de utilizador que servem como teste de funcionalidade da aplicação.

2.1 Interface exibindo a aplicação SOS Mindelo e o seu ícone na tela principal do emulador, juntamente com outras aplicações instaladas.

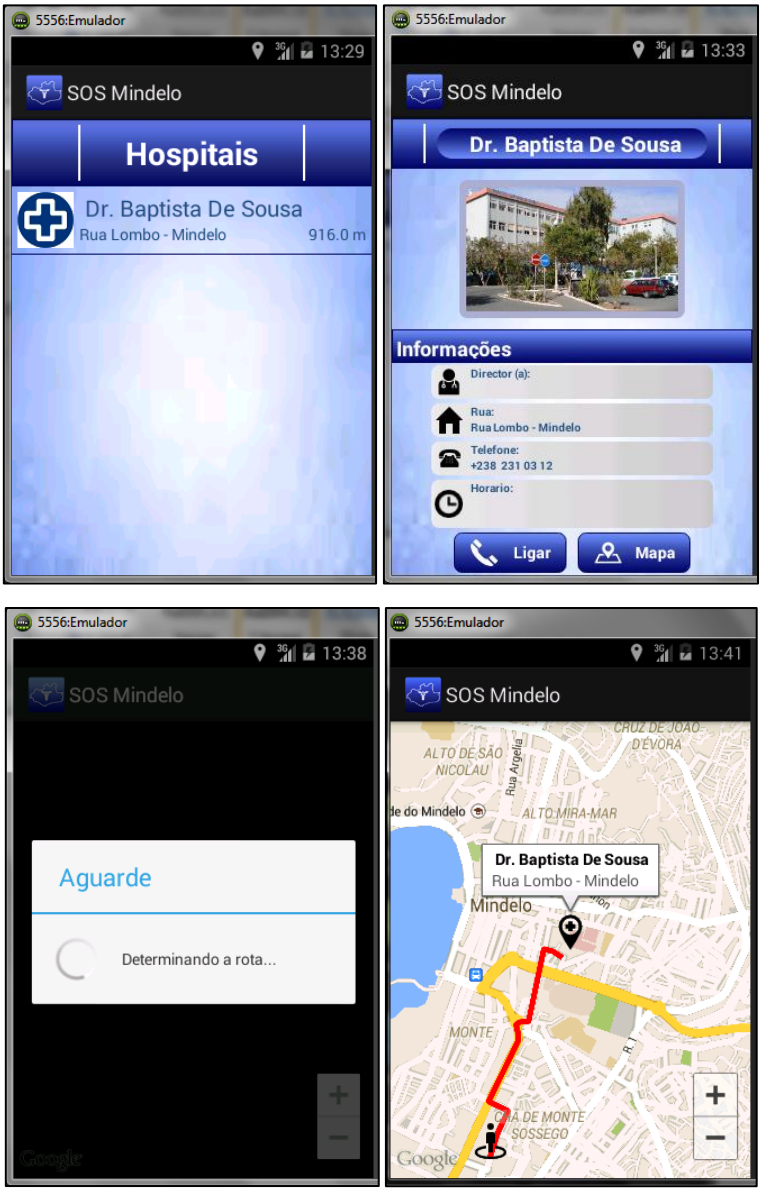


2.2 Interface principal notificando ao utilizador que não existe a ligação com a internet. No entanto esta notificação só aparece quando não há conexão com a internet

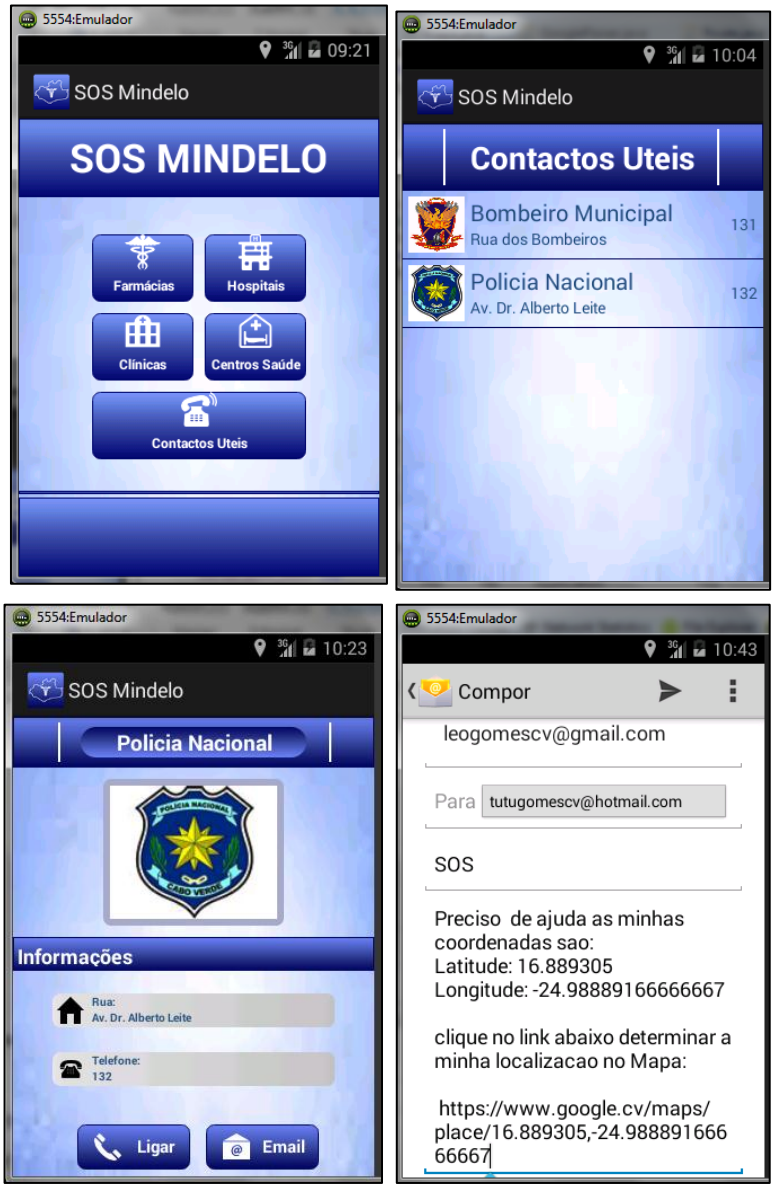
2.3 Interfaces de utilizador que mostra as funcionalidades da aplicação quando é escolhido a opção Farmácias.



2.4 Interface de utilizador que mostra as funcionalidades da aplicação quando é seleccionado a opção Hospitais.



2.5 Interface de utilizador que mostra as funcionalidades da aplicação quando é seleccionado a opção Contactos Úteis e mostra também a funcionalidade de enviar um email.



2.6 Interface principal alertando ao utilizador para activar o GPS do dispositivo para poder obter as coordenadas geográficas. Mas apenas aparece quando o GPS do aparelho estiver desactivado.